Maxime HERVÉ

Aide-mémoire de statistique appliquée à la biologie

*Construire son étude et analyser les résultats à l'aide du logiciel **R***

Version 5(2) (2014)

AVANT-PROPOS

Les phénomènes biologiques ont cela de charmant (et d'extrêmement agaçant) qu'ils sont systématiquement variables. De fait, un biologiste peut donc difficilement se passer des statistiques. Alors, quitte à devoir s'y frotter, autant savoir ce que l'on fait.

L'objectif de cet aide-mémoire est de vous guider dans votre démarche statistique, depuis la construction du protocole expérimental jusqu'à l'analyse des résultats qui en découlent. Il doit vous permettre de vous en sortir seul, tout en assurant une analyse appropriée et rigoureuse. Bien entendu, il ne vous dispense pas de vous poser des questions et il est toujours nécessaire d'adapter un minimum le code proposé à vos propres données. Pour imager les choses, considérez que vous apprenez à faire du vélo et que ce document est la paire de roulettes qui vous évite de chuter. C'est rassurant, mais n'oubliez pas qu'avant tout c'est vous qui pédalez.

Peu de connaissances sont requises pour utiliser cet aide-mémoire, à la fois en statistiques et en langage **R** *– celui que j'ai choisi d'utiliser. Par contre, il exige que vous soyez actif(ve) dans la démarche,* i.e. *que vous identifiiez vous-même là où vous voulez aller. Si vous jouez le jeu, les choses devraient bien se passer.*

*Je vous invite sincèrement à m'envoyer un e-mail (*mx.herve@gmail.com*) si vous trouvez qu'un point n'est pas clair, qu'un autre mériterait d'être ajouté ou approfondi, ou encore qu'une erreur se glisse dans le document.*

Certaines des fonctions présentées dans cet ouvrage nécessitent d'installer des packages qui ne sont pas fournis avec la distribution de base de **R**. *Parmi ceux-ci se trouve le package* `RVAideMemoire`, *qui contient des fonctions que j'ai écrites spécialement pour accompagner cet aide-mémoire. Son développement est donc intimement lié à celui de ce document, et là encore je vous encourage à me faire part de vos remarques, suggestions, critiques et/ou corrections. Cette version 5(2) de l'aide-mémoire correspond aux versions du package* `RVAideMemoire` $\geq$ *0.9-35.*

Enfin, si vous souhaitez être au courant de la sortie de nouvelles versions de l'aide-mémoire ou du package `RVAideMemoire`, *envoyez-moi un message que je vous inscrive sur la liste de diffusion.*

J'espère sincèrement que ce document comblera vos attentes et qu'il vous permettra de vous sentir moins seul dans le monde pas si cauchemardesque des statistiques.

Le 16 mars 2014

Maxime HERVÉ

SOMMAIRE

L'ouvrage est divisé en cinq parties :

Un rappel sur les bases du fonctionnement de R : ce document n'est pas à proprement parler une initiation à R, cependant quelques rappels sont proposés sur la création et l'utilisation des objets qu'il faut maîtriser.

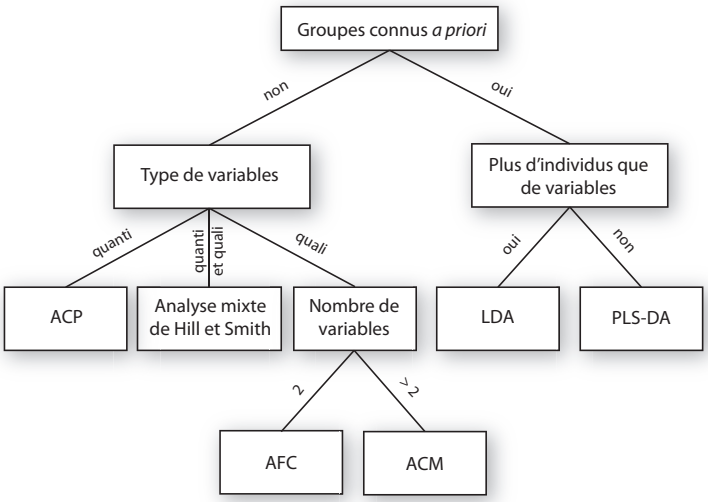
La préparation de l'étude : souvent trop peu d'importance y est attachée. Pourtant, cette phase est au moins aussi cruciale que l'analyse des résultats puisqu'elle détermine la façon dont ceux-ci vont pouvoir être analysés. Une étude bien préparée facilite grandement l'exploitation des résultats, tandis qu'une étude mal préparée entraîne généralement des complications au moment de l'analyse et de l'interprétation.

La préparation et l'importation des données : cette étape apparemment simple peut poser problème par manque d'expérience. Elle est pourtant cruciale, puisque des données mal structurées ou mal importées dans R peuvent conduire à une analyse complètement faussée (ou le plus souvent à des erreurs).

L'analyse descriptive des résultats : ce type d'analyse est toujours indispensable, et selon l'objectif de l'étude il peut être suffisant. L'analyse descriptive est souvent négligée pour « foncer sur les tests », ce qui conduit à oublier la réalité des données (et par conséquent à compliquer voire fausser l'interprétation des résultats).

L'analyse inférentielle des résultats : ce type d'analyse regroupe la détermination des intervalles de confiance et la réalisation des tests statistiques. L'analyse inférentielle est la seule phase de l'étude qui est facultative. Dans tous les cas elle doit passer après l'analyse descriptive.

4.3. Données à plus de deux dimensions



17. L'Analyse en Composantes Principales (ACP)
18. L'Analyse Factorielle des Correspondances (AFC)
19. L'Analyse des Correspondances Multiples (ACM)
20. L'Analyse mixte de Hill et Smith
21. L'Analyse Discriminante Linéaire (LDA) – aspects descriptifs *(Remarque : LDA et AFD (Analyse Factorielle Discriminante) sont des synonymes)*
22. La régression PLS discriminante (PLS-DA) – aspects descriptifs
23. Les classifications automatiques

5. ANALYSE INFÉRENTIELLE DES RÉSULTATS

5.1. Quelques bases théoriques

5.1.1. Lois de probabilité

5.1.1.1. Lois de probabilité discontinues

24. Les lois de probabilité discontinues – généralités
25. La loi binomiale
26. La loi de Poisson
27. La loi binomiale négative

5.1.1.2. Lois de probabilité continues

28. Les lois de probabilité continues – généralités
29. La loi normale
30. La loi exponentielle
31. La loi de χ^2
32. La loi de Fisher - Snedecor
33. La loi de Student

5.1.2. Risques et puissance associés aux tests statistiques

34. Principe des tests statistiques et risques associés à la conclusion
35. Le risque ou seuil de rejet α
36. La correction du seuil de rejet α (ou des *p-values*)
37. Le risque β et la puissance du test
38. Calculer la puissance d'un test *a posteriori*

5.2. Identification et suppression des données aberrantes

39. L'identification et la suppression des données aberrantes

5.3. Intervalles de confiance et erreur standard

40. L'intervalle de confiance et l'erreur standard

41. Tracer un diagramme en barres avec barres d'erreur

5.4. Tests d'hypothèses

42. Les différents types de test statistique

5.4.1. Conditions préalables à l'utilisation des tests

Ces conditions ne sont pas toujours à remplir, cela dépend du test que l'on souhaite utiliser.

43. Le caractère aléatoire et simple d'une série de données

44. L'ajustement à une distribution théorique

45. L'homogénéité des variances de plusieurs séries de données

46. Les transformations de variable

5.4.2. Réalisation des tests

Souvent, plusieurs tests peuvent être utilisés pour répondre à la même question. Les conditions de leur emploi sont cependant plus ou moins restrictives, et leur puissance plus ou moins grande (un test plus restrictif étant généralement plus puissant ; voir fiche **37**). Lorsque plusieurs tests sont disponibles, un arbre de décision est présenté pour aider à choisir le test approprié. Il s'appuie à la fois sur la vérification des conditions à remplir pour utiliser tel ou tel test, ainsi que sur la « qualité » de ces tests (notamment en terme de puissance).

5.4.2.1. La gestion des modèles

Beaucoup d'analyses statistiques passent par l'utilisation de modèles, c'est pourquoi il est important d'en comprendre le fonctionnement global. Il est très fortement conseillé de lire les fiches **47** à **50**.

47. La démarche d'utilisation des modèles

48. La construction de la formule d'un modèle

49. La vérification de la validité d'un modèle

50. L'application d'un test sur un modèle

51. Les comparaisons multiples

52. La sélection de modèle

Statistique uni–, bi– ou multivariée ?

Il est en pratique assez simple de s'orienter :

1. si l'on a *une variable à expliquer*, dont on cherche à savoir si ses valeurs sont influencées par une ou plusieurs variable(s) explicative(s) : statistique *univariée*
2. si l'on étudie la relation entre *deux variables du même type* (quantitatives ou qualitatives), sans qu'il y ait de variable à expliquer et de variable explicative : statistique *bivariée*
3. si l'on a *plusieurs variables à expliquer conjointement* : statistique *multivariée* (attention, si le but de l'analyse est d'expliquer l'appartenance à un groupe – il n'y a donc qu'une variable à expliquer, qualitative – le cadre est bien celui d'une analyse multivariée).

Comment choisir entre analyser plusieurs variables à expliquer séparément ou conjointement ? Cela dépend avant tout des questions auxquelles l'étude doit répondre, mais aussi de la corrélation entre ces variables (voir fiche **76** pour estimer celle-ci) :

– si elle est faible (< 0,4), il n'y a pas d'intérêt à les analyser conjointement
– si elle est modérée (> 0,4 et < 0,7), l'analyse conjointe est très intéressante car elle prend en compte cette corrélation et permet de faire ressortir des effets faibles individuellement mais qui s'additionnent lorsque les variables sont prises ensemble

– si elle est forte (> 0,7), il y a fort à parier que ces variables à expliquer mesurent le même phénomène. Le plus pertinent dans ce cas est de supprimer de l'analyse les variables très corrélées avec les autres, et suivant la corrélation qui lie celles qui restent de choisir entre une analyse séparée ou conjointe.

5.4.2.2. Statistique univariée

Probabilité de réponse, effectif, proportion, moyenne, note, temps de survie ?

Pour savoir sur quel type de test s'orienter, il faut identifier la *nature de la variable à expliquer* :

1. si elle est binaire (*i.e.* seulement deux valeurs possibles) : tests sur des *probabilités de réponse*
2. si elle ne prend que des valeurs entières, positives ou nulles, sans limite (« il y a tant d'individus ») : tests sur des *effectifs*
3. si elle ne prend que des valeurs entières, positives ou nulles, avec une limite maximale (« il y a tant d'individus parmi tant ») : tests sur des *proportions*
4. si elle peut – théoriquement – prendre une infinité de valeurs, entières ou non : tests sur des *moyennes*
5. si elle représente un rang dans un classement (1/2/3, A/B/C, bon/moyen/mauvais…) : tests sur des *notes*. Si le classement n'est composé que de deux notes alors la variable est binaire (on est donc dans le cas 1)
6. si elle représente un temps avant la survenue d'un évènement : tests sur des *temps de survie*. On appelle ces variables à expliquer ce cette façon car les méthodes statistiques associées ont été définies dans le cadre d'analyses de survie, mais elles s'appliquent à toutes les mesures représentant une durée précédant un évènement donné.

Les tests proposés sont systématiquement divisés en deux groupes : les tests « simples » (essentiellement non paramétriques), qui exigent peu de connaissances en statistiques pour être utilisés ; les tests paramétriques basés sur un modèle statistique, qui font appel à des notions un peu plus avancées (tout en restant appréhendables, c'est le but de ce document que de rendre les choses simples ! Voir à ce propos les fiches **47** à **50**). Les modèles sont objectivement plus puissants et permettent bien plus de choses que les tests non paramétriques, et à ce titre ils sont l'approche à privilégier *a priori* lorsque plusieurs alternatives sont disponibles. On pourra toutefois s'orienter vers les tests simples si les conditions d'utilisation du modèle ne sont pas respectées, ou si l'on souhaite délibérément utiliser ce type de test.

1. Tests sur des probabilités de réponse

☛ Tests simples (un facteur et éventuellement des séries appariées, pas de covariable)

Le test de conformité d'une ou de plusieurs probabilité(s) de réponse avec une ou plusieurs valeur(s) théorique(s) est une démarche identique à celle du test de conformité de proportion(s).

53. Comparaison de plusieurs probabilités de réponse

☛ Tests basés sur un modèle statistique (une ou plusieurs variables explicatives, de tous types)

54. Analyser des probabilités de réponse

2. Tests sur des effectifs

☛ Tests simples (sans facteur ni séries appariées ni covariable)

55. Conformité de plusieurs effectifs avec des valeurs théoriques

56. Comparaison de plusieurs effectifs – sans répétition

☛ Tests basés sur un modèle statistique (une ou plusieurs variables explicatives, de tous types)

57. Analyser des effectifs

3. Tests sur des proportions

☛ Tests simples (un facteur et éventuellement des séries appariées, pas de covariable)

58. Conformité d'une proportion avec une valeur théorique
59. Conformité de plusieurs proportions avec des valeurs théoriques
60. Comparaison de deux proportions – sans répétition
61. Comparaison de plus de deux proportions – sans répétition

☛ Tests basés sur un modèle statistique (une ou plusieurs variables explicatives, de tous types)

62. Analyser des proportions

4. Tests sur des moyennes

☛ Tests simples (un à deux facteurs et éventuellement des séries appariées, pas de covariable)

63. Conformité d'une moyenne avec une valeur théorique
64. Comparaison de deux moyennes
65. Comparaison de plus de deux moyennes – un facteur
66. Comparaison de plus de deux moyennes – deux facteurs

☛ Tests basés sur un modèle statistique (une ou plusieurs variables explicatives, de tous types)

On distingue deux situations ici, selon que la relation entre la variable à expliquer et la (ou les) covariable(s) est linéaire ou non (dans ce dernier cas, on suppose qu'il n'y a qu'une covariable et que l'équation qui lie celle-ci et la variable à expliquer est connue). Si toutes les variables explicatives sont des facteurs, le cadre est celui d'une analyse linéaire.

67. Analyser des moyennes – relation(s) linéaire(s)
68. Analyser des moyennes – relation non linéaire

5. Tests sur des notes

☛ Tests simples (un facteur et éventuellement des séries appariées, pas de covariable)

69. Conformité d'une note avec une valeur théorique
70. Comparaison de deux notes
71. Comparaison de plus de deux notes – un facteur

☛ Tests basés sur un modèle statistique (une ou plusieurs variables explicatives, de tous types)

72. Analyser des notes

6. Tests sur des temps de survie

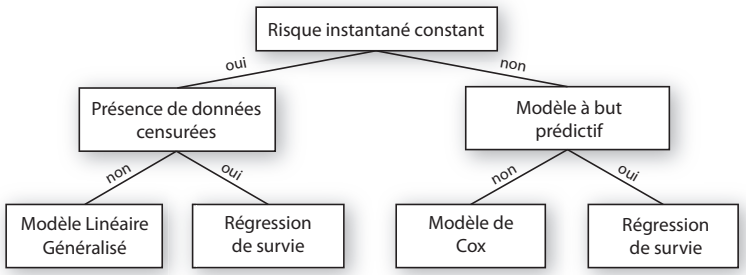
☛ Tests basés sur un modèle statistique (une ou plusieurs variables explicatives, de tous types)

Quel modèle d'analyse de temps de survie choisir ?

Avant toute analyse de temps de survie, il est indispensable d'avoir bien compris les notions suivantes :

- *censure* : un individu est dit censuré lorsque sa mort est survenue avant le début de l'étude (censure *à gauche*) ou quelle n'a pas été observée avant la fin de l'étude (parce que l'étude s'est arrêtée ou parce que l'individu en est sorti ; censure *à droite*). Seules les censures à droite sont traitées ici. Une condition essentielle pour la prise en compte des données censurées dans l'analyse est qu'elles doivent être *indépendantes* des conditions d'expérience
- *risque instantané* : ce risque est celui de mourir à l'instant t, sachant que la mort n'est pas survenue avant. Il peut être constant, ou au contraire augmenter ou diminuer avec le temps.

Plusieurs modèles assez différents peuvent être utilisés pour analyser des temps de survie. Pour choisir le plus approprié :



Pour savoir si le risque instantané est constant, tracer la courbe de survie des individus *via* `plotsurvivors(mort,censure)` (fonction du package `RVAideMemoire`) où `mort` est un vecteur contenant le délai avant la mort de chaque individu et `censure` un vecteur indiquant si l'individu est censuré ou non (0 si censuré ou 1 si non censuré, *i.e.* 0 si la mort n'a pas été observée ou 1 si elle l'a été), dans le même ordre que `mort`. Le risque instantané est considéré comme constant si les points de la courbe de survie sont à peu près alignés sur une droite.

La régression de survie et le modèle de Cox sont des modèles spécifiquement adaptés pour analyser des temps de survie. Les Modèles Linéaires Généralisés (GLMs) sont eux très généraux, l'analyse de temps de survie n'en est qu'un cas particulier.
73. Analyser des temps de survie – Modèle Linéaire Généralisé
74. Analyser des temps de survie – Régression de survie
75. Analyser des temps de survie – Modèle de Cox

5.4.2.3. Statistique bivariée

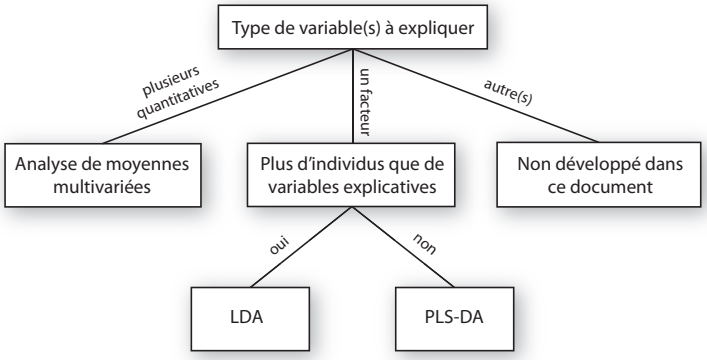
1. Deux variables quantitatives

76. Intensité de la liaison entre deux variables quantitatives
77. Conformité d'un coefficient de corrélation linéaire avec une valeur théorique
78. Comparaison de plusieurs coefficients de corrélation linéaire
79. La régression linéaire simple au sens des moindres rectangles

2. Deux variables qualitatives

80. Association entre deux variables qualitatives

5.4.2.4. Statistique multivariée



81. Analyser des moyennes multivariées
82. L'Analyse Discriminante Linéaire (LDA) – aspects inférentiels *(Remarque : LDA et AFD (Analyse Factorielle Discriminante) sont des synonymes)*
83. La régression PLS discriminante (PLS-DA) – aspects inférentiels

ANNEXES
Bibliographie et ouvrages/documents/liens recommandés

1. Les vecteurs

Le vecteur est à la fois l'objet le plus simple et le plus fondamental du langage R. Il se crée grâce à la fonction `c()`, qui prend comme arguments les éléments du vecteur. Tous ces éléments doivent être du même type : valeurs numériques, chaînes de caractères ou encore niveaux d'un facteur.

EXEMPLE(S)

Pour créer un vecteur numérique :

```
> vecteur <- c(7,9,4,12,18)
> vecteur
[1] 7 9 4 12 18
```

Pour créer un vecteur de chaînes de caractères :

```
> vecteur <- c("H","C","I","G","F")
> vecteur
[1] "H" "C" "I" "G" "F"
```

Pour créer un facteur :

```
> vecteur <- factor(c("niv1","niv2","niv2","niv3","niv1"))
> vecteur
[1] niv1 niv2 niv2 niv3 niv1
Levels: niv1 niv2 niv3
```

Il existe des fonctions ou des abréviations qui permettent de simplifier la création de certains vecteurs usuels :

EXEMPLE(S)

```
> c(1:10)
[1] 1 2 3 4 5 6 7 8 9 10
> seq(from=1,to=3,by=0.25)
[1] 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00
> LETTERS[1:5]
[1] "A" "B" "C" "D" "E"
```

Pour accéder au(x) $i^{\text{ème(s)}}$ élément(s) d'un vecteur, écrire `vecteur[i]`, où `i` peut être une valeur unique ou lui-même un vecteur :

EXEMPLE(S)

```
> vecteur <- seq(from=2,to=16,by=2)
> vecteur
[1] 2 4 6 8 10 12 14 16
> vecteur[5]
[1] 10
> vecteur[c(2,5,8)]
[1] 4 10 16
> vecteur[-c(2,5,8)]
[1] 2 6 8 12 14
```

2. Les tableaux

Les tableaux sont simplement un moyen de regrouper (en colonnes) des vecteurs dans le même objet. Chaque colonne est ici *indépendante*. L'unique contrainte est que tous les vecteurs doivent avoir *la même longueur*.

Pour créer un tableau, utiliser la fonction `data.frame()`, qui prend en arguments les différentes colonnes (de gauche à droite). On peut préciser le titre des colonnes. Dans le cas d'un vecteur de chaînes de caractères, celui-ci est automatiquement transformé en facteur lorsqu'il est intégré au tableau.

EXEMPLE(S)

```
> variable1 <- c(1:5)
> variable2 <- LETTERS[1:5]
> tableau <- data.frame(variable1,variable2)
> tableau
  variable1 variable2
1         1         A
2         2         B
3         3         C
4         4         D
5         5         E
```

Le tableau peut être créé directement :

```
> tableau <- data.frame(variable1=c(1:5),variable2=LETTERS[1:5])
```

Pour accéder à un (ou plusieurs) élément(s) d'un tableau, le principe est le même que pour les vecteurs (voir fiche **1**) excepté qu'il n'y a pas une mais *deux* dimensions à l'objet (*i.e.* les lignes et les colonnes). Le principe d'indexation est valable pour tous les objets à deux dimensions et est celui-ci : `tableau[ligne(s),colonne(s)]`, où `ligne(s)` et `colonne(s)` sont soit des valeurs uniques, soit des vecteurs. Si rien n'est mis avant la virgule toutes les lignes sont sélectionnées, si rien n'est mis après toutes les colonnes sont sélectionnées.

EXEMPLE(S)

```
> tableau[c(1,3),]
  variable1 variable2
1         1         A
3         3         C
> tableau[c(3,5),2]
[1] C E
Levels: A B C D E
```

Dans le cas particulier de la sélection d'une colonne entière, il y a deux autres possibilités :

- `tableau$colonne` où `colonne` est le *nom* de la colonne
- `tableau[,"colonne"]` où `colonne` est le *nom* de la colonne, entre guillemets.

3. Installer et charger un package

Installer un package

Il est nécessaire d'être connecté à internet pour installer un package, car celui-ci doit être téléchargé depuis un serveur. L'installation ne se fait qu'*une seule fois*.

Si R est utilisé depuis la console R, taper `install.packages("package")` où `package` est le nom du package désiré, entre guillemets. Il est demandé ensuite de choisir un serveur, Austria est celui qui est actualisé le plus rapidement (mais Lyon 1 est aussi très bien).

Si R est utilisé depuis la console système, la procédure se fait en deux étapes :

1. télécharger les sources du package à partir de son site de dépôt, le site principal étant le CRAN (*the Comprehensive R Archive Network*) : *http ://cran.r-project.org* rubrique *Packages*
2. installer le package en tapant `R CMD INSTALL package` où `package` est le nom du fichier `tar.gz` contenant les sources.

La procédure expliquée ici est la plus simple, mais il existe de nombreuses variantes. Voir la R FAQ pour plus d'informations : *http ://cran.r-project.org/doc/manuals/R-admin.html#Installing-packages*.

Charger un package

Le chargement d'un package doit se faire *à chaque session où il doit être utilisé*. La commande est simple : `library(package)` où `package` est le nom du package, sans guillemets.

Mettre à jours les packages installés

Pour mettre à jour automatiquement tous les packages installés (chargés ou non), taper `update.packages(ask=FALSE)`. R télécharge alors toutes les mises à jour et les installe.

Il est recommandé de mettre régulièrement à jour ses packages afin de profiter de leur évolution, souvent rapide.

4. Citer R et ses packages

Lors de l'écriture d'un document scientifique, il est une évidence de citer ses sources bibliographiques. Il doit également en être une de citer les logiciels utilisés lors de la réalisation de l'étude. R est certes gratuit, mais il n'en reste pas moins que des dizaines de personnes s'impliquent dans son développement, et qu'il est normal de faire honneur à leur travail en les citant.

R doit être cité dès lors qu'il est utilisé. Pour savoir comment le citer, il suffit de taper `citation()` et de recopier ce qui figure après `To cite R in publications use:`.

Concernant les packages, la règle implicite est de citer tous ceux qui ne sont pas chargés au démarrage de R. Cela comprend les packages installés avec R mais non chargés automatiquement, ainsi que ceux installés par l'utilisateur. Pour savoir comment les citer, taper `citation("package")` où `package` est le nom du package, entre guillemets. Recopier ce qui figure après `To cite the xxx package in publications use:`.

5. Les différents types de variable

Une variable est dite aléatoire si l'on ne peut pas prédire à coup sûr la valeur que prendra un individu. Il existe deux types de variable aléatoire :

1. quantitatives : leurs valeurs représentent une *grandeur*, quantifiable et le plus souvent associée à une unité de mesure. On peut effectuer des opérations mathématiques avec ces variables. Elles peuvent être de deux types :
   - continues, lorsqu'elles peuvent prendre une infinité de valeurs (dans un intervalle donné) : masse, temps, distance, volume…
   - discrètes, lorsqu'elles ne peuvent prendre que certaines valeurs (dans un intervalle donné) : nombre d'individus, d'évènements… Ces variables sont liées le plus souvent à des processus de comptage (où les valeurs prises ne peuvent être qu'entières et positives ou nulles)
2. qualitatives : leurs valeurs ne représentent pas une quantité mais une *catégorie*. On ne peut donc pas effectuer d'opérations mathématiques avec ces variables. On les appelle des *facteurs*, et les valeurs qu'elles peuvent prendre des *classes*, *niveaux* ou *modalités*. Les variables qualitatives peuvent être de deux types :
   - ordinales, lorsque les classes peuvent être ordonnées : rang dans un classement, degré de satisfaction…
   - nominales, lorsque les classes ne peuvent pas être ordonnées : sexe, pays…

Les variables qualitatives peuvent être codées numériquement, mais il est très important de les différencier des variables quantitatives (car elles sont traitées différemment lors de l'analyse statistique). Un principe simple pour faire la différence : si l'on peut remplacer les valeurs d'une variable numérique par des énoncés, alors elle est qualitative. Par exemple, si l'on juge l'état d'une plante par une note allant de 0 à 5, on peut remplacer la note chiffrée par une appréciation du type « mauvais », « satisfaisant »… Cette variable est donc qualitative. Pour ne pas prendre de risque, il est conseillé de ne jamais coder les variables qualitatives numériquement.

Il existe deux types de facteur :

- à effet fixe (« facteur fixe ») : un facteur est fixe si ses classes ont été délibérément choisies, et si le but de l'étude est de les comparer. Par exemple, si l'on veut comparer la taille des individus entre trois espèces, le facteur « espèce » est fixe (à trois classes)
- à effet aléatoire (« facteur aléatoire ») : un facteur est aléatoire si ses classes ont été choisies parmi un grand nombre de classes possibles, et si le but de l'étude n'est pas de les comparer mais simplement de prendre en compte la variabilité qu'il existe entre elles. Par exemple, si les mesures de taille des trois espèces sont réalisées par deux personnes différentes, on peut considérer un facteur « expérimentateur », aléatoire. L'objectif ici n'est en effet pas de comparer les mesures réalisées par les deux personnes, mais de prendre en compte le fait que la façon de réaliser les mesures peut varier entre les deux.

Il y a deux choses à bien garder à l'esprit : (i) la décision de déclarer un facteur comme fixe ou aléatoire est *fondamentale* pour l'analyse des données, car ce ne sont pas les mêmes analyses qui sont réalisées dans les deux cas ; (ii) cette décision doit être prise *selon l'objectif de l'étude, i.e.* la question à laquelle l'étude doit répondre, car aucun facteur n'est fixe ou aléatoire dans l'absolu. Il est donc indispensable de bien réfléchir avant de déclarer un facteur comme fixe ou aléatoire.

Que ce soit pour des variables qualitatives ou quantitatives, si certaines mesures ne sont pas indépendantes entre elles, elles constituent des *séries appariées*. Le cas le plus simple est celui où plusieurs mesures sont réalisées sur un même individu (par exemple avant et après un traitement). Mais d'autres cas plus subtils peuvent se présenter : si des mesures sont réalisées sur des individus apparentés (ces mesures ne sont pas indépendantes car il existe une corrélation d'origine génétique entre elles), si des séries de mesures sont réalisées à des localisations différentes (ces mesures ne sont pas indépendantes car chaque série est influencée par l'environnement local) ou encore si des séries de mesures sont réalisées à des temps différents (ces mesures ne sont pas indépendantes car chaque série est influencée par ce qu'il a pu se passer avant). Il est très important d'identifier les séries appariées lorsqu'elles existent, car des analyses statistiques adaptées doivent alors être utilisées. Les séries appariées sont le plus souvent identifiées par l'introduction d'un facteur aléatoire. Pour les exemples précédents, on a donc respectivement un facteur « individu », un facteur « famille », un facteur « localisation » et un facteur « moment ».

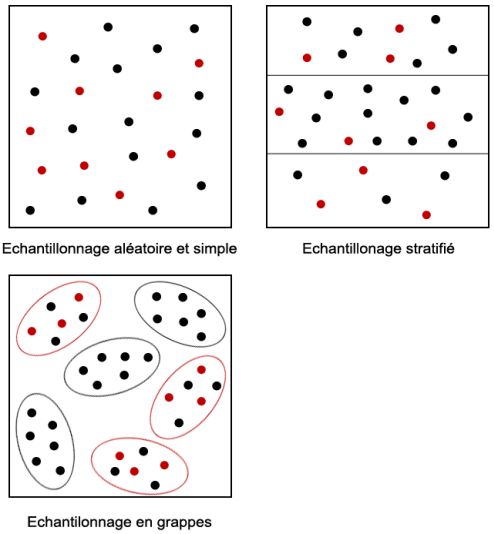
6. Le plan d'échantillonnage

On utilise un plan d'échantillonnage lorsque l'on réalise une étude par enquête, *i.e.* lorsque l'on collecte des informations sur un groupe d'individus dans leur milieu habituel, mais que tous les individus ne sont pas accessibles (par choix ou par contrainte).

Les principales méthodes d'échantillonnage peuvent être regroupées en deux ensembles :

1. l'échantillonnage aléatoire : tous les individus (au sens statistique) ont la même probabilité d'être choisis, et le choix de l'un n'influence pas celui des autres. Différentes méthodes d'échantillonnage aléatoire existent (voir les illustrations) :
   - l'échantillonnage aléatoire et simple : le choix se fait parmi tous les individus de la population (au sens statistique), qui ne forme qu'un grand ensemble
   - l'échantillonnage stratifié : si la population est très hétérogène, elle peut être divisée en sous-ensembles exclusifs (ou strates). Au sein de ces strates l'échantillonnage est ensuite aléatoire et simple. Les strates sont identifiées dans l'analyse statistique comme les niveaux d'un facteur fixe
   - l'échantillonnage en grappes : si les strates sont très nombreuses, on en choisit certaines au hasard (les grappes). Au sein de ces grappes l'échantillonnage est ensuite aléatoire et simple. Les grappes sont identifiées dans l'analyse statistique comme les niveaux d'un facteur aléatoire
   - l'échantillonnage par degrés : il est une généralisation de l'échantillonnage en grappes (qui est en fait un échantillonnage du premier degré). Au sein de la population on choisit des grappes « primaires », puis à l'intérieur de celles-ci des grappes « secondaires » (toujours au hasard), et ainsi du suite... Au dernier niveau l'échantillonnage est aléatoire et simple
2. l'échantillonnage systématique : un premier individu est choisi aléatoirement, puis les autres sont choisis de façon régulière à partir du précédent (dans le temps ou l'espace). L'analyse de ce type d'échantillonnage, qui fait appel à la statistique spatiale ou à l'analyse des séries chronologiques, n'est pas abordée dans cet ouvrage.

Il est important d'identifier la méthode mise en œuvre car les analyses statistiques doivent être adaptées. Seule l'analyse de plans d'échantillonnage aléatoires est abordée dans cet ouvrage.



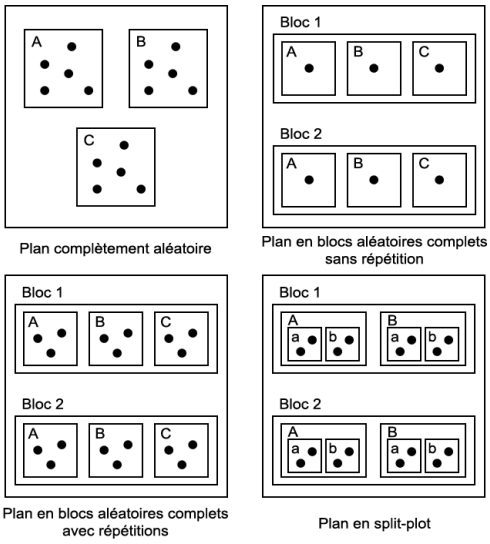
Chaque point représente un individu. Les individus échantillonnés sont représentés en rouge.

7. Le plan d'expérience

On utilise un plan d'expérience lorsque l'on réaliste une étude par expérimentation, *i.e.* lorsque l'on provoque volontairement les faits à étudier. Le plan d'expérience comprend notamment le(s) facteur(s) à faire varier, le nombre de répétitions à réaliser et le dispositif expérimental à mettre en place. L'association des classes de plusieurs facteurs constitue un *traitement*.

Il existe de nombreux types de dispositif expérimental, dont les principaux sont (voir les illustrations) :
- le plan d'expérience complètement aléatoire : chaque individu (au sens statistique) est affecté à un traitement aléatoirement
- le plan d'expérience en blocs aléatoires complets : s'il y a (ou s'il peut y avoir) une grande hétérogénéité entre les individus, ils sont réunis en groupes aussi homogènes que possibles (ou blocs). Au sein de ces blocs chaque individu est ensuite affecté aléatoirement à un traitement, de manière à ce que tous les traitements soient présents dans chacun des blocs. Les blocs sont identifiés dans l'analyse statistique comme les niveaux d'un facteur aléatoire
- le plan d'expérience en blocs aléatoires incomplets : dans ce cas tous les traitements ne sont pas présents dans chacun des blocs
- le plan d'expérience en split-plot : le principe du split-plot est le plus souvent associé à celui des blocs aléatoires complets. Dans ce cas, dans chacun des blocs sont créés autant de sous-blocs qu'il y a de classes au premier facteur étudié. À chacun de ces sous-blocs est associée une classe. Puis chaque sous-bloc est divisé en autant d'unités qu'il y a de classes au second facteur étudié. À chacun de ces « sous-sous-blocs » est associée une classe. Pour plus de deux facteurs, la situation est plus complexe.

Quelle que soit la méthode employée, elle doit être clairement définie car elle doit être prise en compte dans les analyses statistiques.



Chaque point représente un individu. Chaque carré représente un traitement. Les niveaux du 1er facteur sont notés en majuscule, les niveaux du 2nd facteur sont notés en minuscule.

8. La détermination de la taille de l'échantillon à constituer

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]pwr

Il existe un lien entre le seuil de rejet α du test statistique utilisé (voir fiches **34** et **35**), la puissance de ce test (voir fiche **37**), la différence entre les échantillons pour la variable mesurée et la taille des échantillons. Déterminer la taille de l'échantillon à constituer passe donc par fixer les autres paramètres. Ceci implique deux choses importantes :

- choisir *avant de démarrer l'étude* les types de test qui vont être utilisés (ce qui oblige à bien identifier les questions auxquelles l'étude doit répondre) et leur précision. On considère que lorsque l'objectif de l'étude est de *rejeter* H_0 (voir fiche **34**), la puissance du test doit être d'au moins 80 % ; lorsque l'objectif est de *ne pas rejeter* H_0 (voir fiche **34**), la puissance doit être d'au moins 95 %
- avoir une idée de la variabilité naturelle de la variable mesurée et/ou de la différence minimale à détecter. Ceci passe soit par une étude de la bibliographie, soit par la consultation de spécialistes, soit par la réalisation d'un pré-échantillonnage ou d'une pré-expérience.

Dans R, la fonctions `power()` permet, pour quelques tests paramétriques courants, de calculer l'effectif nécessaire lorsque l'on connaît tous les autres paramètres. Lorsque l'on sait à l'avance que l'on ne sera pas dans les conditions d'utilisation de ces tests, on peut tout de même estimer la puissance de leur équivalent non paramétrique. Celle-ci est vaut en effet environ 95 % de la puissance du test paramétrique équivalent.

Toutes les fonctions décrites sont basées sur le même principe : le paramètre `n` doit avoir comme valeur `NULL` tandis que tous les autres doivent être fixés. Toutes les fonctions considèrent que l'effectif est le même dans les différents groupes à comparer (on parle d'effectifs *équilibrés*). Il est fortement conseillé de toujours prévoir ses plans d'échantillonnage ou d'expérience de cette façon, ce qui (i) facilite l'analyse et (ii) permet d'utiliser la plupart des tests non paramétriques, dont l'équilibre des effectifs est une condition.

Comparaison d'une moyenne avec une valeur théorique (voir fiche 63) ou de deux moyennes (voir fiche 64) – test *t* de Student

```
power.t.test(n,delta,sd,sig.level,power,type)
```

avec :
`n` : effectif (identique pour les deux échantillons dans le cas d'une comparaison de deux moyennes)
`delta` : différence minimale à détecter entre la moyenne de l'échantillon et la moyenne théorique, ou entre les deux moyennes
`sd` : écart-type (identique pour les deux échantillons dans le cas d'une comparaison de deux moyennes), dans l'unité des moyennes
`sig.level` : seuil de rejet α (généralement 0,05)
`power` : puissance minimale du test (0,8 ou 0,95 selon l'objectif du test)
`type` : type de test (`"one.sample"` pour la comparaison d'une moyenne avec une valeur théorique, `"two.sample"` pour la comparaison de deux moyennes, `"paired"` pour la comparaison de deux moyennes en séries appariées).

Comparaison de plus de deux moyennes – analyse de variance à un facteur (voir fiche 67)

```
power.anova.test(groups,n,between.var,within.var,sig.level,power)
```

avec :
`groups` : nombre de modalités à comparer
`between.var` : somme des carrés des écarts intergroupe minimale à détecter
`within.var` : somme des carrés des écarts intragroupe (identique pour toutes les modalités).

Il est dans la pratique très difficile d'estimer *a priori* les sommes des carrés des écarts inter et intragroupe. On peut dans ce cas se rabattre sur la fonction `power.t.test()` qui donne des résultats convenables si les conditions d'utilisation de l'analyse de variance sont remplies.

Comparaison de deux proportions – test du χ^2 d'homogénéité (voir fiche 60)

```
power.prop.test(n,p1,p2,sig.level,power)
```

avec `p1`, `p2` : proportions à détecter comme significativement différentes.

Significativité d'un coefficient de corrélation linéaire de Pearson (voir fiche 76)

```
pwr.r.test(n,r,sig.level,power)
```[1]

avec `r` : coefficient de corrélation minimum à détecter comme significativement différent de 0.

9. La construction du tableau de données

La construction d'un tableau de données correctement structuré est une étape importante de l'étude, car si elle est mal réalisée elle peut mener à des résultats faux, ou le plus souvent à des erreurs une fois dans R.

Cette construction nécessite de se poser une question essentielle : quelles sont les variables prises en compte dans l'étude ? Y répondre implique d'identifier les variables quantitatives et les facteurs, ainsi que les classes des facteurs. Si les choses sont claires, l'analyse statistique le sera également.

D'une manière générale, il est conseillé de toujours construire son tableau de données dans un tableur. Cela permet d'enregistrer le jeu de données dans un fichier externe à R, et donc de toujours pouvoir y revenir puisque R ne modifie pas les fichiers externes (sauf si on le lui demande explicitement).

Dans le tableur, la règle est simple : les individus doivent être placés en *lignes* et les variables en *colonnes*.

Il est conseillé de donner un titre à chaque colonne, qui deviendra le nom de la variable dans R. Il est indispensable cependant de respecter certaines règles : les noms de variable ne doivent contenir ni espace, ni caractère accentué, ni symbole (ceci est une règle pour tous les noms d'objet dans R). Si un nom de variable doit contenir deux mots, ils peuvent être séparés par un point (.) ou un tiret bas (_). Mieux vaut également privilégier les noms courts mais clairs, car une fois dans R taper sans cesse des noms de variable longs est vite fastidieux.

Le tableau de données doit absolument obéir à une autre règle : *aucune case ne doit être vide*. La seule exception possible est celle en haut à gauche si les colonnes ont un titre, auquel cas la 1ère colonne sera comprise par R comme le nom des lignes. S'il manque une donnée pour un individu, il faut se demander d'où elle vient :

- si c'est une donnée inutilisable (mesure ratée, mal retranscrite...), pas de problème. On dit alors qu'on a une « donnée manquante », que l'on doit noter `NA` (pour *Not Available, i.e.* donnée manquante). Le tableur comme R reconnaissent le `NA`, qu'ils interprètent correctement
- si la situation est autre, c'est que le tableau est mal construit et qu'en particulier les variables n'ont pas été bien définies. La réflexion s'impose donc pour identifier les variables et reconstruire un tableau de données.

Il est déconseillé de coder les niveaux d'un facteur avec uniquement des chiffres. R comprendrait cette variable comme numérique (et non comme un facteur), ce qui pourrait sérieusement perturber voire empêcher l'analyse.

Si des analyses dans R doivent se faire uniquement sur un sous-ensemble du tableau de données, ou si pour certaines analyses le tableau de données serait plus facile à utiliser s'il était construit autrement, il est conseillé de construire plusieurs tableaux de données séparés. Il est toujours possible de manipuler le tableau initial dans R pour en extraire une partie ou pour le transformer, mais il est clairement plus facile (et surtout moins source d'erreur) lorsque l'on n'a pas l'habitude de le faire en amont, dans le tableur.

10. L'importation du tableau de données dans R

Il existe de nombreuses méthodes pour importer ses données dans R. Une seule est présentée ici, qui est à la fois très simple, fonctionne dans la plupart des situations et peut être utilisée sur toutes les plateformes.

La procédure se fait en trois étapes :

1. dans le tableur, sélectionner toutes les cases constituant le tableau de données
2. copier ce tableau dans le bloc-notes et enregistrer le fichier en format `.txt`
3. dans R, charger le tableau de données grâce à la fonction `read.table()` et le stocker dans un objet : `tableau<-read.table("fichier")` où `fichier` est le nom du fichier texte (avec l'extension `.txt` et éventuellement le chemin qui mène à ce fichier), entre guillemets.

R étant un logiciel anglo-saxon, le séparateur décimal qu'il utilise est le point. Or dans les tableurs français (et donc dans le fichier texte) le séparateur décimal est la virgule. Si le tableau de données contient des valeurs décimales, il est donc nécessaire de préciser à R qu'il interprète la virgule comme séparateur décimal. Ajouter pour cela l'argument `dec=","` à la fonction `read.table()`.

Si les colonnes du tableau de données ont un titre, qui doit donc être interprété comme le nom de la variable, ajouter l'argument `header=TRUE`.

Une fois le tableau importé, il est indispensable de vérifier qu'il n'y a pas eu d'erreur pendant son chargement. Pour cela appeler le résumé du tableau *via* `summary(tableau)`. R renvoie un résumé de chaque variable :

- pour une variable numérique, R donne des indications sur sa distribution : minimum, 1er quartile, médiane, moyenne, 3ème quartile et maximum
- pour un facteur, R donne le nombre d'individus par classe.

Si un facteur est codé numériquement (par exemple un facteur binaire ou un facteur ordinal), R l'interprète comme une variable numérique. Pour transformer cette variable en facteur, taper `tableau$variable<-factor(tableau$variable)` où `variable` est le nom de la variable.

11. Les graphiques de dispersion

Ce type de graphique représente toutes les données individuelles d'un vecteur, d'une matrice ou d'un tableau. Il permet d'avoir un aperçu de la variabilité des données et d'identifier les observations aberrantes. Son tracé est basé sur la fonction `stripchart()`.

Pour représenter un vecteur : `stripchart(vecteur)`.

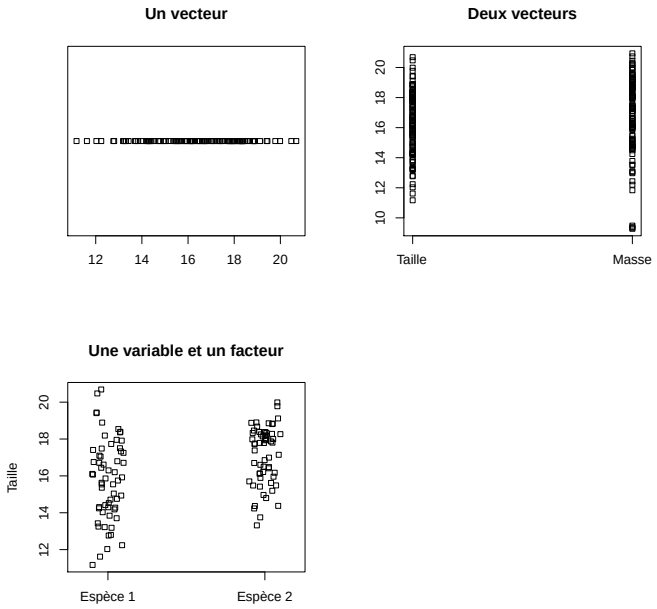
Pour représenter plusieurs vecteurs : `stripchart(list(vecteur1,vecteur2,...))`.
Pour donner un nom aux vecteurs sur le graphe, ajouter l'argument `group.names=c("Nom1","Nom2",...)`.

Pour représenter des données en fonction d'un facteur : `stripchart(donnees~facteur)` où les deux objets sont des vecteurs contenant la valeur de chaque individu (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Pour représenter les données verticalement, ajouter l'argument `vertical=TRUE`.

Pour que les valeurs identiques ne se superposent pas, ajouter l'argument `method="jitter"` (par défaut `method="overplot"`).

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.
Pour modifier la légende de l'axe horizontal, utiliser l'argument `xlab="Légende"`.
Pour modifier la légende de l'axe vertical, utiliser l'argument `ylab="Légende"`.
Pour (beaucoup) plus d'options graphiques, voir `?par`.



12. Les histogrammes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Ce type de graphique divise les données contenues dans un vecteur en classes, et représente chaque classe en effectif ou densité. Il permet d'avoir un aperçu de la distribution des données. Son tracé est basé sur la fonction `hist()`.

Pour représenter les classes en effectif : `hist(vecteur)`.
Pour représenter les classes en densité : `hist(vecteur,freq=FALSE)` (`freq=TRUE` par défaut, ce qui représente les effectifs).

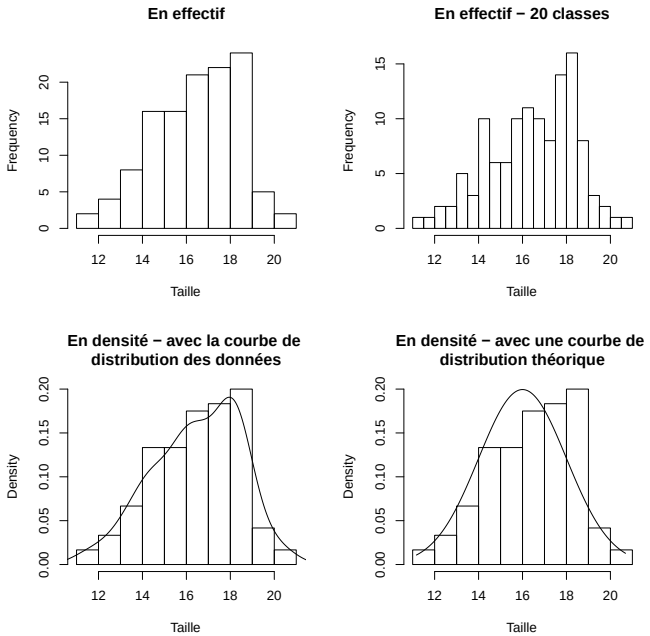
Pour ajouter une courbe de densité : `lines(density(vecteur))`. Une telle courbe ne peut être ajoutée que sur un histogramme tracé en densité.

Pour ajouter une courbe de distribution théorique : `lines(seq2(vecteur)`[1]`,dloi(seq2(vecteur)`[1]`,par))` où `loi` est la loi de probabilité choisie et `par` ses paramètres séparés par une virgule (voir fiches **24** à **33**).

Pour modifier le nombre de classes, ajouter l'argument `breaks=n` où `n` est le nombre de coupures souhaitées (il y a donc `n` + 1 classes).

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.
Pour modifier la légende de l'axe horizontal, utiliser l'argument `xlab="Légende"`.
Pour modifier la légende de l'axe vertical, utiliser l'argument `ylab="Légende"`.
Pour (beaucoup) plus d'options graphiques, voir `?par`.

Pour tracer l'histogramme d'une variable par niveau d'un facteur : `byf.hist(variable~facteur)`[1] où `variable` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».



13. Les boîtes à moustaches

Ce type de graphique représente de façon simplifiée la dispersion des données contenues dans un vecteur. Il permet d'avoir un aperçu de la distribution et de la variabilité des données, et d'identifier les observations aberrantes. Son tracé est basé sur la fonction `boxplot()`.

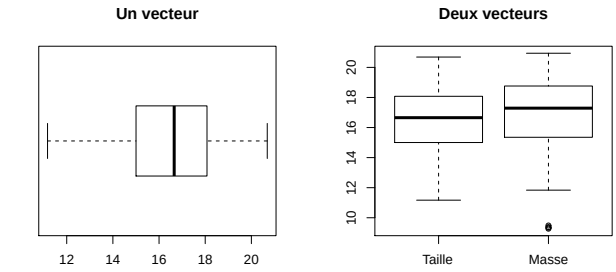
Pour représenter un vecteur : `boxplot(vecteur)`. Le trait épais représente la médiane, la boîte est formée par les valeurs des 1er et 3ème quartiles, et les moustaches mesurent maximum 1,5 fois la longueur de l'interquartile (*i.e.* la différence 3ème quartile - 1er quartile). Les valeurs au-delà des moustaches sont représentées individuellement.

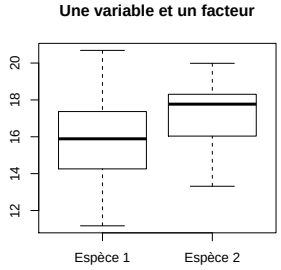
Pour représenter plusieurs vecteurs : `boxplot(list(vecteur1,vecteur2,...))`.
Pour donner un nom aux boîtes, ajouter l'argument `names=c("Nom1","Nom2",...)`.

Pour représenter une boîte par niveau d'un facteur : `boxplot(variable~facteur)` où `variable` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Pour représenter les boîtes horizontalement, ajouter l'argument `horizontal=TRUE`.

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.
Pour modifier la légende de l'axe horizontal (si les boîtes sont tracées horizontalement), utiliser l'argument `xlab="Légende"`.
Pour modifier la légende de l'axe vertical (si les boîtes sont tracées verticalement), utiliser l'argument `ylab="Légende"`.
Pour (beaucoup) plus d'options graphiques, voir `?par`.





14. La réduction des données à une dimension

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Les paramètres suivants permettent de réduire une série de données (*i.e.* un vecteur) à quelques valeurs apportant une information générale.

Paramètres de position

Ils permettent de donner un ordre de grandeur des données.
Moyenne : `mean(vecteur)` où `vecteur` est un vecteur contenant la valeur de chaque individu.
Médiane : `median(vecteur)`.
Mode : `mod(vecteur)`[1].

Si les vecteurs contiennent des données manquantes (`NA`), ajouter l'argument `na.rm=TRUE` aux fonctions `mean()` et `median()`. La fonction `mod()` gère par défaut les données manquantes.

Paramètres de dispersion

Ils permettent d'estimer la variabilité des données autour des paramètres de position.
Variance : `var(vecteur)`.
Écart-type (*standard deviation*) : `sd(vecteur)`.
Coefficient de variation : `cv(vecteur)`[1]. Le coefficient est par défaut exprimé en valeur absolue et en pourcentage.

Si les vecteurs contiennent des données manquantes (`NA`), ajouter l'argument `na.rm=TRUE` aux fonctions `var()` et `sd()`. La fonction `cv()` gère par défaut les données manquantes.

Les fonctions `var()` et `sd()` calculent la variance et l'écart-type *non biaisés* (*i.e.* sur la base de $n-1$ et non n, n étant l'effectif de l'échantillon). La fonction `cv()` est basée sur l'écart-type non biaisé.

Pour calculer la valeur d'un paramètre (de position ou de dispersion) par niveau d'un facteur, utiliser `tapply(vecteur,facteur,function(x) fonction)` où `vecteur` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre), et `fonction` la fonction à utiliser. Dans cette fonction, `vecteur` doit être remplacé par `x` : `mean(x)`, `var(x,na.rm=TRUE)`, `cv(x)`...

— EXEMPLE(S) —

```
> variable <- c(1:60)
> facteur <- factor(rep(LETTERS[1:3],each=20))
> tapply(variable,facteur,function(x) mean(x))
   A    B    C
10.5 30.5 50.5
```

Avec un deuxième facteur, la fonction renvoie une matrice :

```
> facteur2 <- factor(rep(letters[1:2],30))
> tapply(variable,list(facteur2,facteur),function(x) mean(x))
   A  B  C
a 10 30 50
b 11 31 51
```

Le premier facteur défini les *lignes* de la matrice, le second en définit les *colonnes*.

15. Les nuages de points

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Ce type de graphique permet de représenter les valeurs de deux variables numériques pour chaque individu, dans un graphe du type $y = f(x)$. Il permet d'avoir un aperçu de la liaison qui peut exister entre ces variables. Il peut être utilisée avec deux vecteurs, une matrice à deux colonnes ou un tableau à deux colonnes. Son tracé est basé sur la fonction `plot()`.

Pour représenter deux vecteurs `x` et `y` contenant la valeur de chaque individu pour les deux variables (dans le même ordre) : `plot(y~x)`. Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.
Pour modifier la légende de l'axe horizontal, utiliser l'argument `xlab="Légende"`.
Pour modifier la légende de l'axe vertical, utiliser l'argument `ylab="Légende"`.
Pour (beaucoup) plus d'options graphiques, voir `?par`.

Pour ajouter une droite du type $y = ax + b$: `abline(b,a)`.

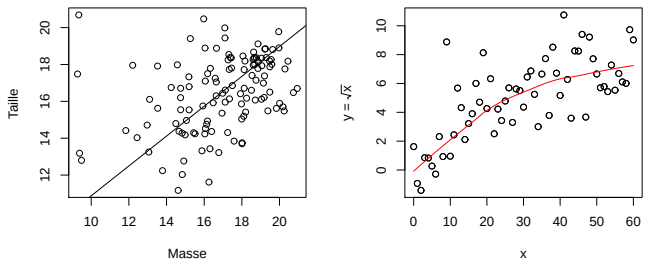
Pour ajouter une droite de régression linéaire au sens des moindres carrés (voir fiche **67**) : `abline(lm(y~x))`.

Pour ajouter une droite de régression linéaire au sens des moindres rectangles (voir fiche **79**) : `abline(least.rect(y~x)`[1]`)`.

Pour ajouter une courbe de tendance du nuage de points : `panel.smooth(x,y)`.

Pour ajouter une droite horizontale : `abline(h=ordonnee)`.

Pour ajouter une droite verticale : `abline(v=abscisse)`.



16. La réduction des données à deux dimensions

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Les paramètres suivants permettent de réduire deux séries de données à quelques valeurs apportant une information générale sur la liaison qui peut les unir.

Pour deux vecteurs `x` et `y` contenant la valeur de chaque individu pour chaque série (dans le même ordre) :
Covariance : `cov(x,y)`.
Coefficient de corrélation linéaire de Pearson (paramétrique ; voir fiche **76**) : `cor(x,y,method="pearson")`.
Coefficient de corrélation linéaire de Spearman (non paramétrique ; voir fiche **76**) : `cor(x,y,method="spearman")`.

Les vecteurs doivent avoir la même taille (*i.e.* contenir autant de valeurs). S'ils contiennent des données manquantes (`NA`), ajouter l'argument `use="complete.obs"` aux fonctions `cov()` et `cor()`. Seuls les couples de données complets sont alors considérés.

La régression linéaire au sens des moindres carrés (voir fiche **67**) s'utilise dans le cas où la variable `y` (dite *dépendante* ou *à expliquer*) varie en fonction de la variable `x` (dite *indépendante* ou *explicative*). Pour récupérer les paramètres de la droite : `lm(y~x)$coefficients` (le symbole `~` signifie « expliqué par » ou « en fonction de »). La première valeur (`Intercept`) correspond à l'ordonnée à l'origine, la seconde au coefficient directeur de la droite. La fonction `lm()` construit un modèle linéaire reliant `x` et `y` au sens des moindres carrés.

La régression linéaire au sens des moindres rectangles (voir fiche **79**) s'utilise dans le cas où les deux variables sont considérées sur un pied d'égalité (elles sont dites *interdépendantes*). Pour récupérer les paramètres de la droite : `least.rect(y~x)$coefficients`[1]. La première valeur (`Intercept`) correspond à l'ordonnée à l'origine, la seconde au coefficient directeur de la droite. La fonction `least.rect()`[1] construit un modèle linéaire reliant `x` et `y` au sens des moindres rectangles.

Les vecteurs `x` et `y` doivent avoir la même taille pour les deux fonctions `lm()` et `least.rect()`[1]. Ils peuvent contenir des données manquantes (`NA`).

17. L'Analyse en Composantes Principales (ACP)

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]ade4

Préparation des données

Les données doivent être contenues dans un tableau avec en colonnes les variables (avec un titre) et en lignes les individus. La 1ère case du tableau (en haut à gauche) doit être vide si la 1ère colonne contient le nom des individus. Donner un nom aux individus est conseillé pour faciliter l'interprétation de l'analyse.

Le tableau ne doit pas contenir de données manquantes (`NA`). Pour supprimer les lignes qui en possèdent une fois dans R : `tableau<-na.omit(tableau)`, où `tableau` est le tableau de départ.

Lancement de l'analyse

L'ACP se prête mieux à l'analyse de relations linéaires et de variables ayant une distribution symétrique. Pour observer ces caractères, utiliser `pairs(tableau,panel=panel.smooth)` d'une part et `hist()` (voir fiche **12**) avec chacune des variables d'autre part. Si besoin transformer les variables (voir fiche **46** ; en pratique on utilise souvent la transformation logarithmique).

L'ACP est créée *via* `ACP<-dudi.pca(tableau)`[1]. Lorsque cette commande est passée, R renvoie le graphe des *valeurs propres* (ou *pouvoirs de synthèse*) associées à toutes les *variables de synthèse* (ou *axes* ou *composantes principales*) possibles, et demande le nombre d'axes à sélectionner. La valeur propre d'un axe est proportionnelle à la quantité d'information (du tableau initial) représentée par cet axe. Le choix du nombre de composantes à retenir est toujours subjectif et repose sur un compromis entre un nombre faible d'axes (au-delà de trois l'interprétation devient très difficile) et une quantité d'information représentée importante.

Qualité de la synthèse

On évalue la qualité de l'analyse par le pourcentage de variance expliquée par les axes retenus. En ACP cette variance correspond à la variance *totale* qui existe dans les données initiales. Ces pourcentages de variance sont obtenus *via* `inertia.dudi(ACP)`[1]. Dans le tableau renvoyé (`$TOT`), les pourcentages sont cumulés dans la colonne `ratio`.

Représentations graphiques

Graphe des individus

Le graphe est obtenu *via* `s.label(ACP$li)`[1]. Le tableau `$li` de l'objet `ACP` contient les coordonnées des individus dans les différents plans factoriels. Pour sélectionner un plan factoriel, ajouter les arguments `xax=num1` et `yax=num2` où `num1` est le numéro du 1er axe choisi et `num2` celui du 2nd (il faut avoir sélectionné au moins deux axes au départ de l'ACP). Par convention on choisit pour l'axe horizontal celui des deux expliquant le plus de variance.

Pour l'interprétation, plus deux individus sont éloignés sur le graphe et plus ils le sont dans le tableau initial.

Pour ajouter comme information supplémentaire un facteur définissant des groupes d'individus :

– sous forme d'ellipses : `s.class(dfxy=ACP$li,fac=facteur)`[1]
– sous forme de polygones de contour : `s.chull(dfxy=ACP$li,fac=facteur,optchull=1)`[1]

où `facteur` est un vecteur contenant la modalité de chaque individu. Pour donner une couleur à chaque groupe ajouter l'argument `col=couleur` où `couleur` est un vecteur contenant la couleur de chaque modalité du facteur, dans l'ordre alphabétique des modalités. Le plan factoriel peut être précisé grâce aux arguments `xax` et `yax`.

Pour superposer les graphes obtenus avec les fonctions `s.label()`[1] et `s.class()`[1]/`s.chull()`[1], utiliser d'abord `s.label()`[1] puis `s.class()`[1]/`s.chull()`[1] avec l'argument `add.plot=TRUE`.

Graphe des variables

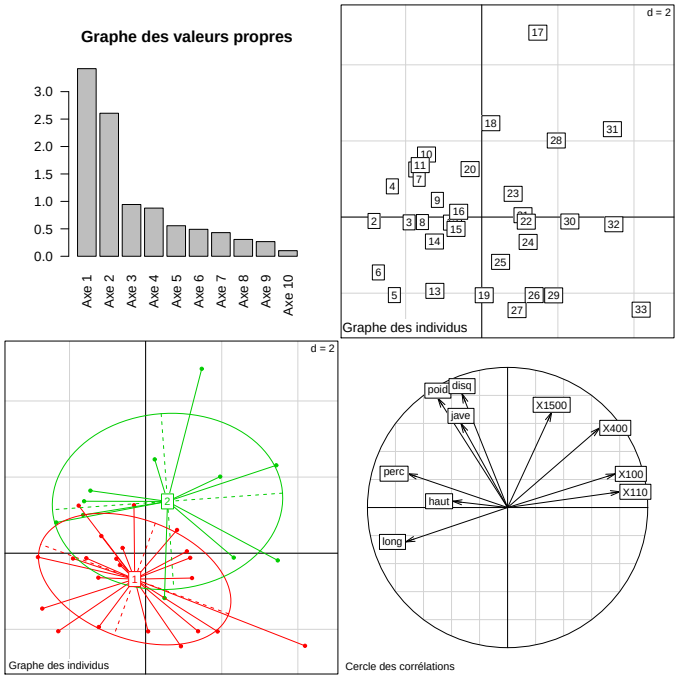
Pour interpréter les axes et comprendre ce qui différencie les individus, tracer le cercle des corrélations *via* `s.corcircle(ACP$co)`[1]. Le tableau `$co` de l'objet `ACP` contient les coefficients de corrélation (de Pearson, voir fiche **76**) qui relient chaque variable d'origine avec chaque axe. Le plan factoriel peut être sélectionné grâce aux arguments `xax` et `yax`.

Sur le graphe, plus une flèche est longue et mieux elle est représentée par les deux axes. Plus sa direction est proche de celle d'un axe et mieux elle est représentée (*i.e.* corrélée) par cet axe. L'angle entre deux flèches indique la corrélation qui existe entre les variables : angle aigu = positive ; angle droit = nulle, angle obtus = négative.

Pour l'interprétation, se concentrer sur les flèches les mieux représentées (*i.e.* les plus longues). En lien avec le graphe des individus, plus un individu est situé vers l'avant de la flèche (lorsqu'on le projette sur la direction de la flèche) et plus il possède une valeur élevée pour cette variable (et vice-versa).

Un 1er axe très corrélé de façon positive avec toutes les variables est souvent le signe d'un « effet taille ». Il convient dans ce cas de se placer dans des plans factoriels ne comprenant pas le 1er axe pour l'interprétation.

Pour représenter à la fois les individus et les variables sur le même graphe, utiliser `scatter(ACP,posieig="none")`[1]. Pour ne pas afficher le numéro (ou le nom) des individus, ajouter l'argument `clab.row=0`. L'échelle des flèches sur cette double représentation est arbitraire, elle peut être changée sans que cela ne change l'interprétation.



18. L'Analyse Factorielle des Correspondances (AFC)

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`ade4`

Préparation des données

Les données doivent être en effectifs et organisées dans un tableau de contingence, du type :

		Facteur B		
		Classe 1	...	Classe c
Facteur A	Classe 1			
	...			
	Classe k			

Ce tableau peut-être créé tel quel dans le tableur (*i.e.* en amont de R), auquel cas il faut laisser la 1ère case (en haut à gauche) vide pour que la 1ère colonne soit bien comprise comme le nom des lignes (*i.e.* les classes du facteur A). Il peut également être obtenu à partir d'un tableau au format classique *via* `tableau<-table(facteurA,facteurB)` où `facteurA` et `facteurB` sont des vecteurs contenant la valeur de chaque individu pour chaque facteur (dans le même ordre).

Lancement de l'analyse

L'AFC est sensible aux effectifs faibles. Quand ils sont trop nombreux, mieux vaut regrouper les classes pour obtenir des effectifs plus élevés.

L'AFC est créée *via* `AFC<-dudi.coa(tableau)`[1]. Lorsque cette commande est passée, R renvoie le graphe des *valeurs propres* (ou *pouvoirs de synthèse*) associées à toutes les *variables de synthèse* (ou *axes*) possibles, et demande le nombre d'axes à sélectionner. La valeur propre d'un axe est proportionnelle à la quantité d'information (du tableau initial) représentée par cet axe. Le choix du nombre de variables de synthèse à retenir est toujours subjectif et repose sur un compromis entre un nombre faible d'axes (au-delà de trois l'interprétation devient très difficile) et une quantité d'information représentée importante.

Qualité de la synthèse

On évalue la qualité de l'analyse par le pourcentage d'information expliqué par les axes retenus. Ces pourcentages sont obtenus *via* `inertia.dudi(AFC)`[1]. Dans le tableau renvoyé (`$TOT`), les pourcentages sont cumulés dans la colonne `ratio`.

Représentations graphiques

Pour visualiser graphiquement le résultat de l'AFC, utiliser `scatter(AFC,posieig="none")`[1]. Pour sélectionner un plan factoriel, ajouter les arguments `xax=num1` et `yax=num2` où `num1` est le numéro du 1er axe choisi et `num2` celui du 2nd (il faut avoir sélectionné au moins deux axes au départ de l'AFC). Par convention on choisit pour l'axe horizontal celui des deux expliquant le plus d'information.

Pour l'interprétation, la proximité des modalités représente leur liaison plus ou moins forte dans le tableau initial. On voit donc sur le même graphe à la fois les similitudes entre modalités d'un même facteur, et la liaison entre les modalités des deux facteurs.

Pour ajouter comme information supplémentaire un facteur définissant des groupes d'individus :

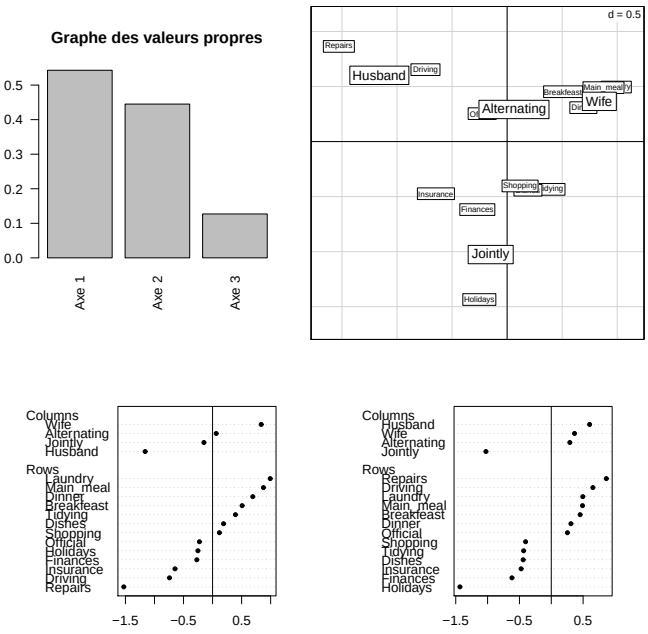
– sous forme d'ellipses : `s.class(dfxy=individus,fac=facteur)`[1]

– sous forme de polygones de contour : `s.chull(dfxy=individus,fac=facteur,optchull=1)`[1]

où `individus` est `AFC$li` si les *lignes* du tableau initial sont considérées comme les individus à regrouper, `AFC$co` si ce sont les *colonnes*, et `facteur` est un vecteur contenant la modalité de chaque individu. Pour donner une couleur à chaque groupe ajouter l'argument `col=couleur` où `couleur` est un vecteur contenant la couleur de chaque modalité du facteur, dans l'ordre alphabétique des modalités. Le plan

factoriel peut être précisé grâce aux arguments `xax` et `yax`.

Pour interpréter les axes il peut être utile d'utiliser `score(AFC,xax=num,dotchart=TRUE)`[1] où `num` est le numéro de l'axe à représenter. Le graphique montre la répartition des modalités des deux facteurs sur l'axe choisi. Utiliser `abline(v=0)` pour ajouter une ligne marquant l'origine de l'axe.



19. L'Analyse des Correspondances Multiples (ACM)

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]ade4

Préparation des données

Les données doivent être contenues dans un tableau avec en colonnes les variables (avec un titre) et en lignes les individus. La 1ère case du tableau (en haut à gauche) doit être vide si la 1ère colonne contient le nom des individus. Donner un nom aux individus est conseillé pour faciliter l'interprétation de l'analyse.

Le tableau ne doit pas contenir de données manquantes (`NA`). Pour supprimer les lignes qui en possèdent une fois dans R : `tableau<-na.omit(tableau)`, où `tableau` est le tableau de départ.

Lancement de l'analyse

L'ACM est sensible aux effectifs faibles. Quand ils sont trop nombreux, mieux vaut regrouper les classes pour obtenir des effectifs plus élevés.

L'ACM est créée *via* `ACM<-dudi.acm(tableau)`[1]. Lorsque cette commande est passée, R renvoie le graphe des *valeurs propres* (ou *pouvoirs de synthèse*) associées à toutes les *variables de synthèse* (ou *axes*) possibles, et demande le nombre d'axes à sélectionner. La valeur propre d'un axe est proportionnelle à la quantité d'information (du tableau initial) représentée par cet axe. Le choix du nombre de variables de synthèse à retenir est toujours subjectif et repose sur un compromis entre un nombre faible d'axes (au-delà de trois l'interprétation devient très difficile) et une quantité d'information représentée importante.

Qualité de la synthèse

On évalue la qualité de l'analyse par le pourcentage d'information expliqué par les axes retenus. Ces pourcentages sont obtenus *via* `inertia.dudi(ACM)`[1]. Dans le tableau renvoyé (`$TOT`), les pourcentages sont cumulés dans la colonne `ratio`.

Représentations graphiques

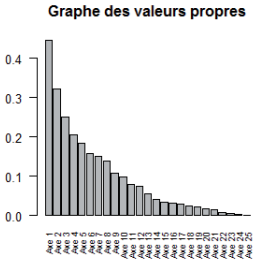
Pour représenter les résultats de l'ACM, utiliser `scatter(ACM)`[1]. Les résultats sont séparés pour chaque variable, toutes représentées sur le même plan factoriel. Pour sélectionner un plan factoriel, ajouter les arguments `xax=num1` et `yax=num2` où `num1` est le numéro du 1er axe choisi et `num2` celui du 2nd (il faut avoir sélectionné au moins deux axes au départ de l'ACM). Par convention on choisit pour l'axe horizontal celui des deux expliquant le plus d'information. Les modalités de chaque variable sont représentées par des ellipses portant leur nom. Pour rendre le graphique plus clair et donner une couleur à chaque modalité, ajouter l'argument `col=couleur` où `couleur` est un vecteur contenant autant de couleurs qu'il y a de modalités possibles.

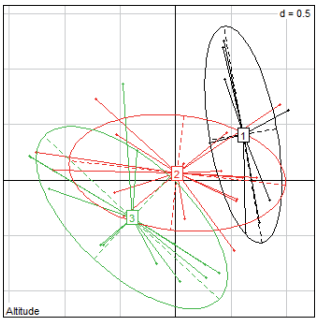
Pour ne représenter qu'une seule variable, le même résultat est obtenu *via* `s.class(dfxy=ACM$li, fac=variable,col=couleur,sub="nom")`[1] où `variable` est la variable choisie et `nom` son nom (entre guillemets). Le plan factoriel peut être sélectionné grâce aux arguments `xax` et `yax`.

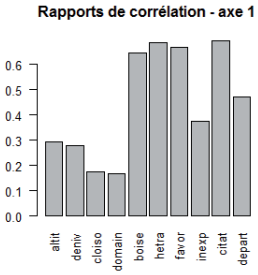
Le tableau `ACM$cr` contient les rapports de corrélation (variant de 0 à 1) entre les variables de départ et les axes. Pour représenter graphiquement ces rapports, utiliser `barplot(ACM$cr[,num],names.arg=rownames(ACM$cr),las=2)` où `num` est le numéro de l'axe à représenter. Pour l'interprétation des axes, se concentrer sur les variables les plus structurantes, *i.e.* dont le rapport de corrélation est le plus proche de 1.

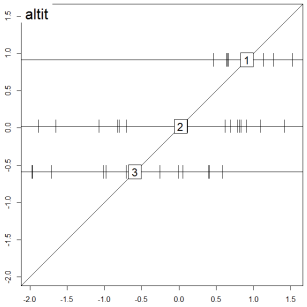
Une aide à l'interprétation est fournie par la fonction `score(ACM,xax=num)`[1] où `num` est le numéro de l'axe à représenter. Le graphique montre la répartition des modalités de chaque variable sur l'axe choisi. Pour sélectionner les variables à représenter, ajouter l'argument `which.var=variables` où `variables` est un vecteur contenant le numéro des variables choisies (*i.e.* le numéro des colonnes correspondantes dans le tableau initial).

L'ACM est clairement une analyse plus difficile à interpréter que l'ACP ou l'AFC. Aussi, il est bon de limiter la difficulté en ne considérant pas des dizaines de variables mais en se limitant aux questions essentielles.


Graphe des valeurs propres
0.4
0.3
0.2
0.1
0.0
Axe 1
Axe 2
Axe 3
Axe 4
Axe 5
Axe 6
Axe 7
Axe 8
Axe 9
Axe 10
Axe 11
Axe 12
Axe 13
Axe 14
Axe 15
Axe 16
Axe 17
Axe 18
Axe 19
Axe 20
Axe 21
Axe 22
Axe 23
Axe 24
Axe 25


d = 0.5
1
2
3
Altitude


Rapports de corrélation - axe 1
0.6
0.5
0.4
0.3
0.2
0.1
0.0
altit
denv
cloiso
domain
boise
hetra
favor
inexp
citat
depart


altit
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
1
2
3
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5

20. L'Analyse mixte de Hill et Smith

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]ade4, [2]RVAideMemoire

Préparation des données

Les données doivent être contenues dans un tableau avec en colonnes les variables (avec un titre) et en lignes les individus. La 1ère case du tableau (en haut à gauche) doit être vide si la 1ère colonne contient le nom des individus. Donner un nom aux individus est conseillé pour faciliter l'interprétation de l'analyse.

Le tableau ne doit pas contenir de données manquantes (`NA`). Pour supprimer les lignes qui en possèdent une fois dans R : `tableau<-na.omit(tableau)`, où `tableau` est le tableau de départ.

Lancement de l'analyse

L'analyse est plus efficaces avec des variables quantitatives ayant une distribution symétrique. Tracer des histogrammes pour observer ce caractère (voir fiche **12**), et si besoin transformer les variables (voir fiche **46** ; en pratique on utilise souvent la transformation logarithmique). Pour ce qui est des variables qualitatives, l'analyse est sensible aux effectifs faibles. Quand ils sont trop nombreux, mieux vaut regrouper les classes pour obtenir des effectifs plus élevés.

L'analyse est réalisée *via* `AMix<-dudi.hillsmith(tableau)`[1]. Lorsque cette commande est passée, R renvoie le graphe des *valeurs propres* (ou *pouvoirs de synthèse*) associées à toutes les *variables de synthèse* (ou *axes*) possibles, et demande le nombre d'axes à sélectionner. La valeur propre d'un axe est proportionnelle à la quantité d'information (du tableau initial) représentée par cet axe. Le choix du nombre de variables de synthèse à retenir est toujours subjectif et repose sur un compromis entre un nombre faible d'axes (au-delà de trois l'interprétation devient très difficile) et une quantité d'information représentée importante.

Qualité de la synthèse

On évalue la qualité de l'analyse par le pourcentage d'information expliqué par les axes retenus. Ces pourcentages sont obtenus *via* `inertia.dudi(AMix)`[1]. Dans le tableau renvoyé (`$TOT`), les pourcentages sont cumulés dans la colonne `ratio`.

Représentations graphiques

Pour représenter les résultats de l'analyse, la fonction `scatter(AMix)`[1] peut être utilisée. Elle représente dans le plan factoriel les individus, les variables quantitatives comme dans une ACP (voir fiche **17**) et les variables qualitatives comme dans une ACM (voir fiche **19**). Pour sélectionner un plan factoriel, ajouter les arguments `xax=num1` et `yax=num2` où `num1` est le numéro du 1er axe choisi et `num2` celui du 2nd (il faut avoir sélectionné au moins deux axes au départ de l'analyse). Par convention on choisit pour l'axe horizontal celui des deux expliquant le plus d'information.

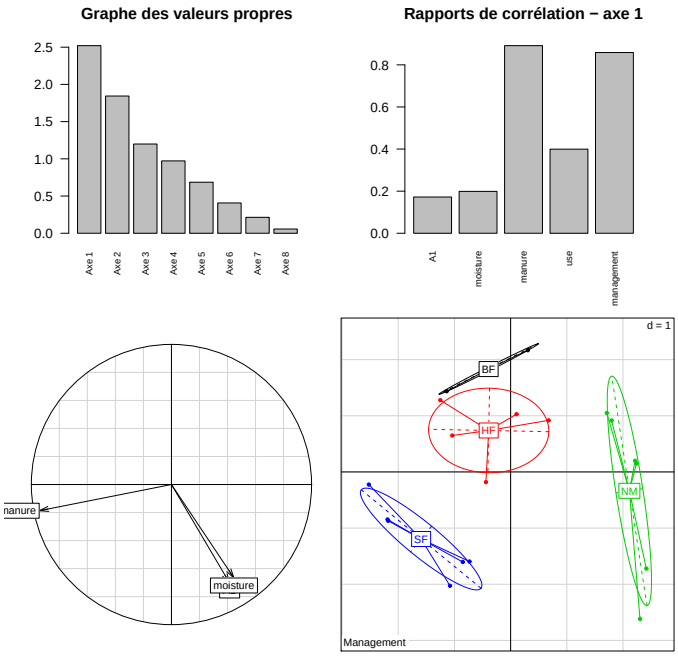
Cette représentation peut cependant être illisible. Si tel est le cas, utiliser les fonctions suivantes :

- `scat.mix.numeric(AMix)`[2], qui représente les variables quantitatives sur un cercle des corrélation (comme en ACP, voir fiche **17**)
- `scat.mix.categorical(AMix)`[2], qui représente les variables qualitatives à la manière d'une ACM (voir fiche **19**).

Le tableau `AMix$cr` contient les rapports de corrélation (variant de 0 à 1) entre les variables de départ et les axes. Pour les représenter graphiquement, utiliser `barplot(AMix$cr[,num],names.arg=rownames(AMix$cr),las=2)` où `num` est le numéro de l'axe à représenter. Pour l'interprétation des axes, se concentrer sur les variables les plus structurantes, *i.e.* dont le rapport de corrélation est le plus proche de 1.

Une aide à l'interprétation est fournie par la fonction `score(AMix,xax=num)`[1] où `num` est le numéro de l'axe à représenter. Là encore la représentation est de type ACP ou ACM selon la nature des variables. Pour sélectionner les variables à représenter, ajouter l'argument `which.var=variables` où `variables`

est un vecteur contenant le numéro des variables choisies (*i.e.* le numéro des colonnes correspondantes dans le tableau initial).



21. L'Analyse Discriminante Linéaire (LDA) – aspects descriptifs

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]MASS, [2]RVAideMemoire, [3]ade4

Préparation des données

Les données doivent être contenues dans un tableau avec en colonnes les variables (avec un titre) et en lignes les individus. La 1ère case du tableau (en haut à gauche) doit être vide si la 1ère colonne contient le nom des individus. Donner un nom aux individus est conseillé pour faciliter l'interprétation de l'analyse. Pour des raisons pratiques, il est conseillé de mettre le facteur (*i.e.* la variable définissant les groupes à discriminer) avant les variables quantitatives.

Le tableau ne doit pas contenir de données manquantes (`NA`). Pour supprimer les lignes qui en possèdent une fois dans **R** : `tableau<-na.omit(tableau)`, où `tableau` est le tableau de départ.

Lancement de l'analyse

La LDA exige théoriquement que les variables explicatives (quantitatives) suivent une distribution normale multivariée et que leurs matrices de variance-covariance soient homogènes. On peut tester graphiquement la 1ère condition (voir fiche **44**), mais il n'existe pas réellement de test fiable pour la 2nde. En pratique la LDA est toutefois robuste au non-respect de ces deux conditions.

La LDA est créée *via* `LDA<-lda(explicatives,facteur)`[1] où `explicatives` correspond aux colonnes du tableau contenant les variables quantitatives et `facteur` est la colonne contenant le facteur. Par défaut tous les axes possibles sont retenus, *i.e.* $n-1$ où n est le nombre de classes du facteur.

— EXEMPLE(S) —

Si le tableau contient cinq colonnes et que le facteur est dans la 1ère :

```
> LDA <- lda(tableau[,2:5],tableau[,1])
```
[1]

Ou encore, si le facteur s'appelle `facteur` :

```
> LDA <- lda(tableau[,2:5],tableau$facteur)
```
[1]

Qualité de la discrimination

On évalue la qualité de l'analyse par deux valeurs :

- le pourcentage de variance expliquée par les axes. En analyse discriminante cette variance correspond à la variance *intergroupe* qui existe dans les données initiales. Ces pourcentages sont obtenus *via* `DA.var(LDA)`[2]
- le pourcentage d'erreur de classification, qui est lié aux aspects inférentiels de la LDA (voir fiche **82**). Il est obtenu *via* `DA.valid(LDA)`[2].

Représentations graphiques

Graphe des individus

Récupérer les coordonnées des individus sur les plans factoriels nécessite une étape préliminaire : `LDA.coord<-LDA.format(LDA)`[2].

Le graphe peut ensuite être obtenu de deux façons selon que l'on souhaite représenter les groupes par des ellipses ou des polygones de contour :

- ellipses : `s.class(LDA.coord$li,fac=facteur)`[3]
- polygones de contour : `s.chull(LDA.coord$li,fac=facteur,optchull=1)`[3]

Dans les deux cas, le tableau `$li` de l'objet `LDA.coord` contient les coordonnées des individus sur les différents plans factoriels. Pour sélectionner un plan factoriel, ajouter les arguments `xax=num1` et `yax=num2` où `num1` est le numéro du 1er axe choisi et `num2` celui du 2nd (il faut qu'il y ait au moins deux axes possibles). Par convention on choisit pour l'axe horizontal celui des deux expliquant le plus de variance. Pour donner une couleur à chaque groupe ajouter l'argument `col=couleur` où `couleur` est un vecteur contenant la couleur de chaque modalité du facteur, dans l'ordre alphabétique des modalités.

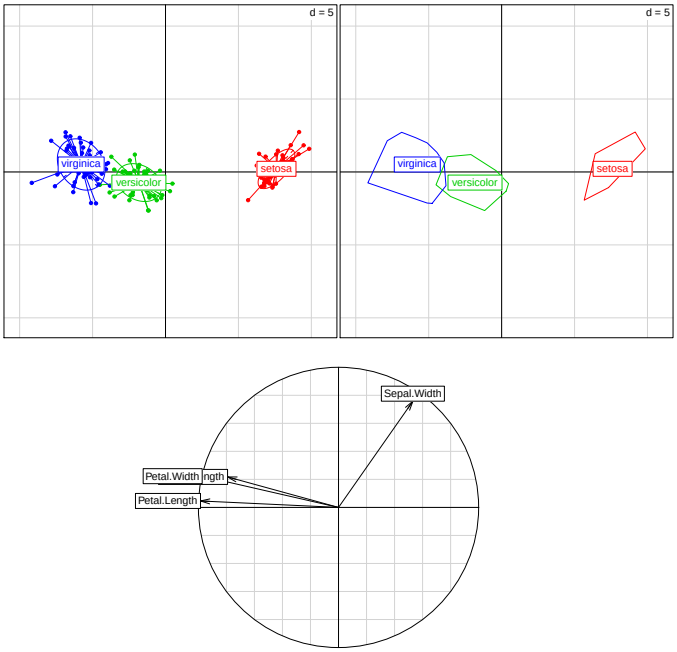
Si le facteur n'a que deux niveaux, la LDA ne se fait que sur un axe. Utiliser dans ce cas `plot1comp.ind(LDA.coord$li,fac=facteur)`[2] pour tracer le graphe des individus.

Graphe des variables

Pour interpréter les axes et comprendre ce qui sépare les groupes, tracer le cercle des corrélations *via* `s.corcircle(LDA.coord$co)`[3]. Le tableau `$co` de l'objet `LDA.coord` contient les coefficients de corrélation (de Pearson, voir fiche **76**) qui relient chaque variable d'origine avec chaque axe. Le plan factoriel peut être sélectionné grâce aux arguments `xax` et `yax`. Si la LDA ne se fait que sur un axe, utiliser `plot1comp.var(LDA.coord$co)`[2].

Sur le graphe, plus une flèche est longue et mieux elle est représentée par l'(es) axe(s). Plus sa direction est proche de celle d'un axe et mieux elle est représentée (*i.e.* corrélée) par cet axe. L'angle entre deux flèches indique la corrélation qui existe entre les variables : angle aigu = positive ; angle droit = nulle, angle obtus = négative.

Pour l'interprétation, se concentrer sur les flèches les mieux représentées (*i.e.* les plus longues). En lien avec le graphe des individus, plus un individu est situé vers l'avant de la flèche (lorsqu'on le projette sur la direction de la flèche) et plus il possède une valeur élevée pour cette variable (et vice-versa).



22. La régression PLS discriminante (PLS-DA) – aspects descriptifs

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`mixOmics`, [2]`RVAideMemoire`, [3]`ade4`

Préparation des données

Les données doivent être contenues dans un tableau avec en colonnes les variables (avec un titre) et en lignes les individus. La 1ère case du tableau (en haut à gauche) doit être vide si la 1ère colonne contient le nom des individus. Donner un nom aux individus est conseillé pour faciliter l'interprétation de l'analyse. Pour des raisons pratiques, il est conseillé de mettre le facteur (*i.e.* la variable définissant les groupes à discriminer) avant les variables quantitatives.

Lancement de l'analyse

La PLS-DA est créée *via* `PLSDA<-plsda(explicatives,facteur)`[1] où `explicatives` correspond aux colonnes du tableau contenant les variables quantitatives et `facteur` est la colonne contenant le facteur. Par défaut deux axes sont retenus, mais cette valeur peut être changée en utilisant l'argument `ncomp=k` où `k` est le nombre d'axes à retenir. En pratique on retient souvent $n-1$ axes, où n est le nombre de classes du facteur. On peut toutefois parfaitement en retenir moins pour faciliter l'interprétation (moyennant, forcément, une perte d'information).

— EXEMPLE(S) —

Si le tableau contient 25 colonnes et que le facteur est dans la 1ère :

```
> PLSDA <- plsda(tableau[,2:25],tableau[,1])[1]
```

Ou encore, si le facteur s'appelle `facteur` :

```
> PLSDA <- plsda(tableau[,2:25],tableau$facteur)[1]
```

Qualité de la discrimination

On évalue la qualité de l'analyse par deux valeurs :

- le pourcentage de variance expliquée par les axes. En analyse discriminante cette variance correspond à la variance *intergroupe* qui existe dans les données initiales. Ces pourcentages sont obtenus *via* `DA.var(PLSDA)`[2]
- le pourcentage d'erreur de classification, qui est lié aux aspects inférentiels de la PLS-DA (voir fiche **83**). Il est obtenu *via* `DA.valid(PLSDA)`[2].

Représentations graphiques

Graphe des individus

Le graphe peut être obtenu de deux façons selon que l'on souhaite représenter les groupes par des ellipses ou des polygones de contour :

- ellipses : `s.class(PLSDA$variates$X,fac=facteur)`[3]
- polygones de contour : `s.chull(PLSDA$variates$X,fac=facteur,optchull=1)`[3]

Dans les deux cas, le tableau `$X` de l'objet `$variates`, lui-même contenu dans l'objet `PLSDA`, contient les coordonnées des individus sur les différents plans factoriels. Pour sélectionner un plan factoriel, ajouter les arguments `xax=num1` et `yax=num2` où `num1` est le numéro du 1er axe choisi et `num2` celui du 2nd (il faut avoir retenus au moins deux axes au début de l'analyse). Par convention on choisit pour l'axe horizontal celui des deux expliquant le plus de variance. Pour donner une couleur à chaque groupe ajouter l'argument `col=couleur` où `couleur` est un vecteur contenant la couleur de chaque modalité du facteur, dans l'ordre alphabétique des modalités.

Graphe des variables

Récupérer les coefficients de corrélations (de Pearson, voir fiche **76**) entre les variables d'origine et les axes nécessite une étape préliminaire : `PLSDA.corr<-cor(PLSDA$X,PLSDA$variates$X,use="pairwise")`.

Pour ensuite interpréter les axes et comprendre ce qui sépare les groupes, tracer le cercle des corrélations *via* `s.corcircle(PLSDA.corr)`[3]. Le plan factoriel peut être sélectionné grâce aux arguments `xax` et `yax`.

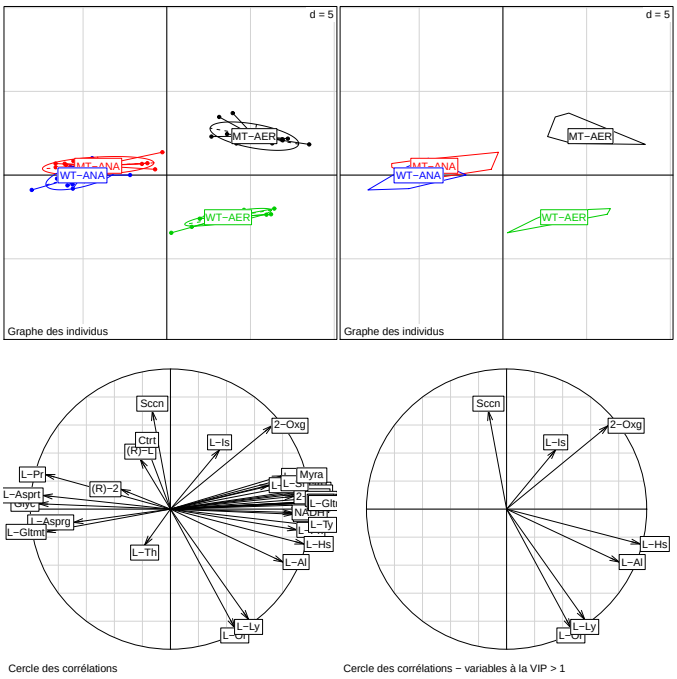
Sur le graphe, plus une flèche est longue et mieux elle est représentée par l'(es) axe(s). Plus sa direction est proche de celle d'un axe et mieux elle est représentée (*i.e.* corrélée) par cet axe. L'angle entre deux flèches indique la corrélation qui existe entre les variables : angle aigu = positive ; angle droit = nulle, angle obtus = négative.

Pour l'interprétation, se concentrer sur les flèches les mieux représentées (*i.e.* les plus longues). En lien avec le graphe des individus, plus un individu est situé vers l'avant de la flèche (lorsqu'on le projette sur la direction de la flèche) et plus il possède une valeur élevée pour cette variable (et vice-versa).

La PLS-DA est souvent utilisée dans un contexte où il y a beaucoup (possiblement plusieurs dizaines) de variables explicatives, ce qui peut rendre très difficile l'interprétation du cercle des corrélations. Il existe cependant un indicateur qui permet de faire le tri dans ces variables pour ne s'intéresser qu'à celles qui sont les plus discriminantes : la VIP (*Variable Importance in the Projection*). Chaque variable explicative possède une valeur de VIP, qui peut globalement aller de 0 à 2. On se concentre généralement sur les variables à la VIP > 1 (mais il ne faut pas oublier que pertinence *statistique* ne veut pas forcément dire pertinence *biologique*).

Pour obtenir ces VIP : `VIP<-PLSDA.VIP(PLSDA)`[2]. Pour n'afficher que les variables à la VIP > 1 sur le cercle des corrélations : `s.corcircle(PLSDA.corr[VIP$sup1,])`[3]. Le vecteur `$sup1` de l'objet `VIP` contient le nom des variables à la VIP > 1.

Attention toutefois, les variables qui sont *statistiquement* les plus intéressantes pour la discrimination ne sont pas nécessairement les seules qui sont *biologiquement* pertinentes. Un examen attentif des relations entre toutes les variables (*via* le cercle des corrélations) est obligatoire.



23. Les classifications automatiques

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Les classifications automatiques sont utilisées pour regrouper des individus en classes, le plus souvent en parallèle d'une ACP (voir fiche **17**) ou d'une AFC (voir fiche **18**). Il existe deux types de classification automatique :

- non hiérarchique : le nombre de classes est déterminé *a priori*
- hiérarchique : les individus sont agrégés successivement pour aboutir à un arbre de classification (appelé *dendrogramme*). C'est à partir de cet arbre que l'on détermine le nombre de classes à générer.

Il existe plusieurs méthodes de classification automatique. Seules deux sont décrites ici : la méthode des *k-means* (non hiérarchique) et la classification ascendante hiérarchique (hiérarchique).

La méthode des *k-means*

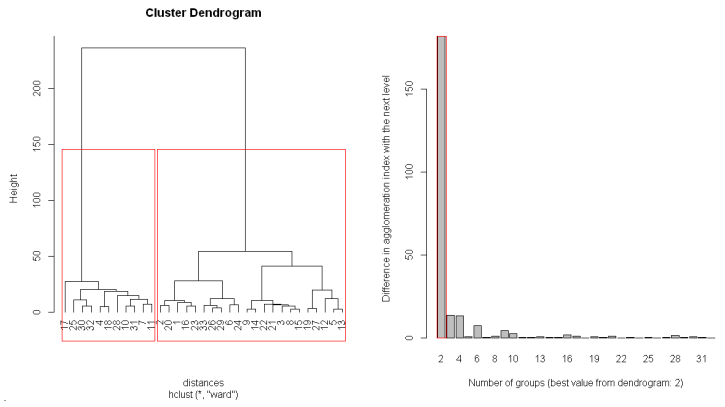
L'obtention de la classe de chaque individu se fait en une seule étape *via* `classes<-factor(kmeans(tableau,centers=nombre)$cluster)`, où `nombre` est le nombre de classes à générer et `tableau` le tableau de données à l'origine de l'analyse multivariée. Attention, les individus à classer sont les *lignes* du tableau de données. Pour considérer les *colonnes* comme individus dans le cas d'une classification parallèle à une AFC (voir fiche **18**), remplacer `tableau` par `t(tableau)` (la fonction `t()` inverse les lignes et les colonnes d'un tableau ou d'une matrice).

La Classification Ascendante Hiérarchique (CAH)

L'obtention de la classe de chaque individu se fait en plusieurs étapes :

1. calculer la *distance* entre chaque individu. En pratique on utilise le plus souvent la distance euclidienne (mais il en existe d'autres) : `distances<-dist(tableau,method="euclidian")`. Si nécessaire, remplacer `tableau` par `t(tableau)`
2. regrouper les individus sur la base d'un *critère d'agglomération*. Il existe plusieurs critères, en pratique on utilise souvent celui de Ward (ce n'est pas le seul, et aucun n'est le meilleur dans l'absolu) : `dendrogramme<-hclust(distances,method="ward")`
3. représenter le dendrogramme : `plot(dendrogramme,hang=-1)`. La distance verticale qui sépare deux groupes est représentative de leur proximité dans le tableau de données. À partir de cet arbre, sélectionner le nombre de classes le plus pertinent. Une aide à la décision est proposée par la fonction `dendro.gp(dendrogramme)`[1].
4. affecter chaque individu à une classe : `classes<-factor(cutree(dendrogramme,k=nombre))`, où `nombre` est le nombre de classes souhaité.

Une fois que le vecteur `classes` a été obtenu, les classes peuvent être représentées graphiquement (voir fiche **17** pour l'ACP, fiche **18** pour l'AFC).

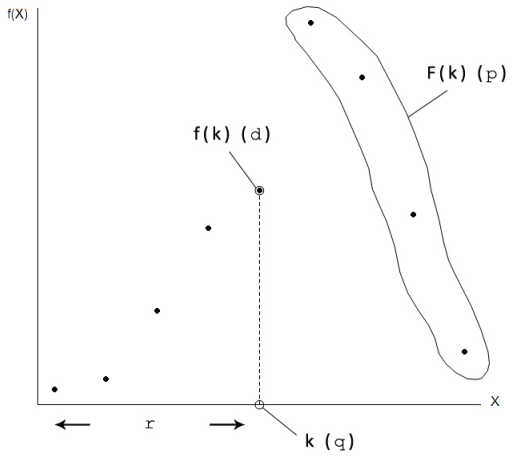


24. Les lois de probabilité discontinues – généralités

Ces lois s'appliquent à des variables quantitatives discrètes.

Paramètres :
- k : chaque valeur possible rencontrée dans la population par la variable discrète X. Également appelée quantile
- $f(k)$: fréquence, ou probabilité, associée à chaque valeur de la variable discrète X. Également appelée distribution de probabilité de X ou fonction de masse de X. Comprise entre 0 et 1
- $F(k)$: somme des probabilités $f(k)$ situées à droite ou à gauche de k, suivant la situation. Également appelée fonction de répartition de X. On note $F(k)_{\text{droite}} = P(X > k)$ et $F(k)_{\text{gauche}} = P(X \leq k)$. Comprise entre 0 et 1.

Dans R :
- `dY()` : donne la probabilité $f(k)$ pour une distribution de type `Y`
- `pY()` : donne la fonction de répartition $F(k)$ pour une distribution de type `Y`. R considère par défaut la répartition à gauche, préciser `lower.tail=FALSE` pour la répartition à droite
- `qY()` : donne la valeur k de la variable X correspondant à une valeur de $F(k)$ pour une distribution de type `Y`. R considère par défaut la répartition à gauche de k, préciser `lower.tail=FALSE` pour la répartition à droite
- `rY()` : donne une série de valeurs aléatoires de la variable X pour une distribution de type `Y`.

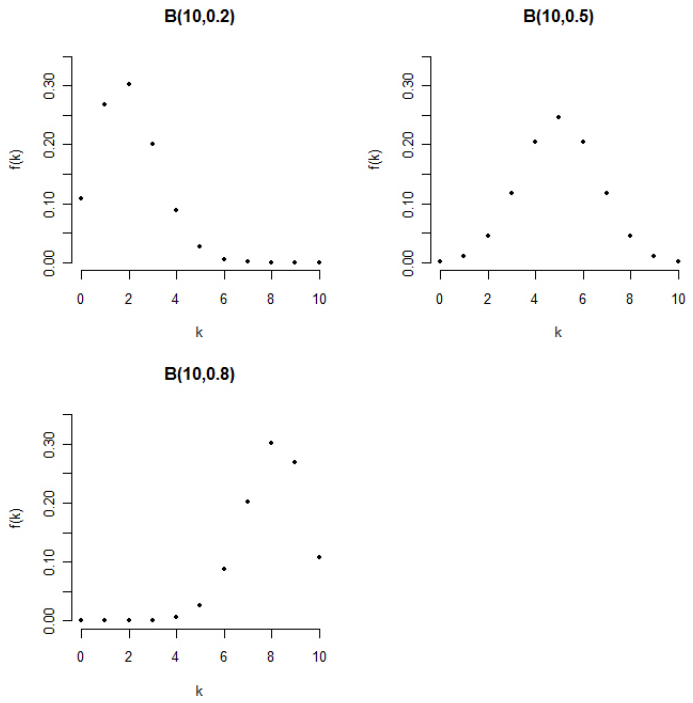


25. La loi binomiale

La loi binomiale est la loi suivie par les résultats de tirages aléatoires lorsqu'il n'y a que deux possibilités mutuellement exclusives de résultat et que la probabilité d'obtenir chaque possibilité est constante au cours de l'expérience (*i.e.* population infinie ou tirages avec remise). La loi donne la probabilité d'obtenir k fois le résultat A quand n tirages sont réalisés.

Écriture : $B(n,p)$
avec :
n : nombre de tirages
p : probabilité associée au résultat A

Dans **R** (voir fiche **24** pour des explications) :
`dbinom(k,n,p)`
`pbinom(k,n,p)`
`qbinom(F(k),n,p)`
`rbinom(x,n,p)` avec `x` : nombre de valeurs à générer



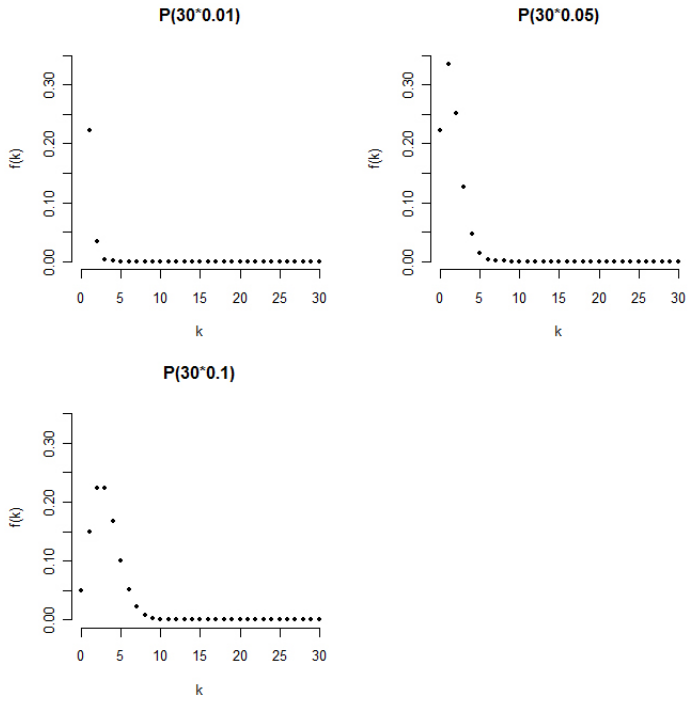
26. La loi de Poisson

La loi de Poisson est une limite de la loi binomiale (voir fiche **25**) quand p tend vers 0 et n vers l'infini (la loi de Poisson est souvent appelée « loi des évènements rares »). L'approximation de la loi binomiale par la loi de Poisson est possible quand quand $p < 0,1$ et $n > 30$.

Sa moyenne est égale à sa variance et vaut np (ou λ).

Écriture : $P(np)$ ou $P(\lambda)$
avec :
n : nombre de tirages
p : probabilité associée au résultat rare

Dans **R** (voir fiche **24** pour des explications) :
`dpois(k,n*p)` ou `dpois(k,lambda)`
`ppois(k,n*p)` ou `ppois(k,lambda)`
`qpois(F(k),n*p)` ou `qpois(F(k),lambda)`
`rpois(x,n*p)` ou `rpois(x,lambda)` avec `x` : nombre de valeurs à générer

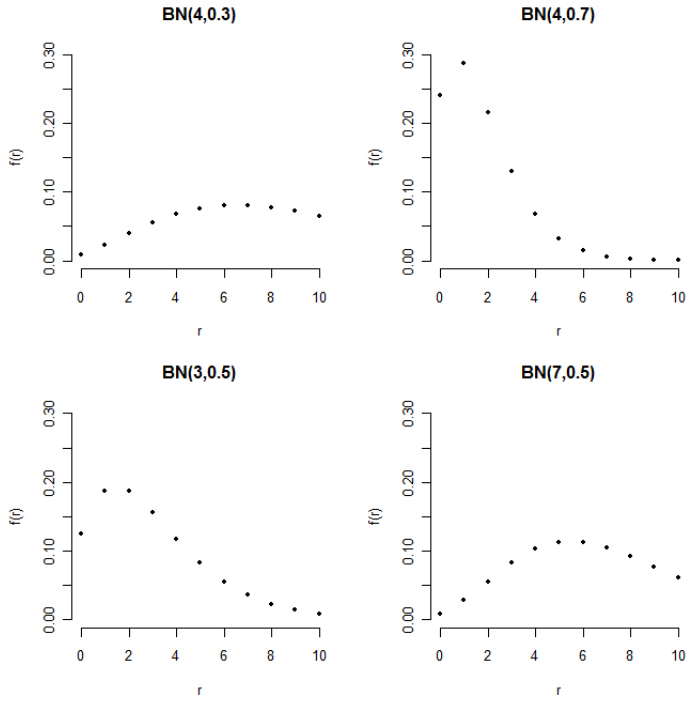


27. La loi binomiale négative

La loi binomiale négative correspond à la même situation que la loi binomiale (voir fiche **25**), mais elle donne la probabilité d'obtenir r résultats B avant d'obtenir k résultats A (approche par l'échec).

Écriture : $BN(k,p)$
avec :
k : nombre de résultats A désirés
p : probabilité associée au résultat A

Dans **R** (voir fiche **24** pour des explications) :
`dnbinom(r,k,p)`
`pnbinom(r,k,p)`
`qnbinom(F(r),k,p)`
`rnbinom(x,k,p)` avec `x` : nombre de valeurs à générer

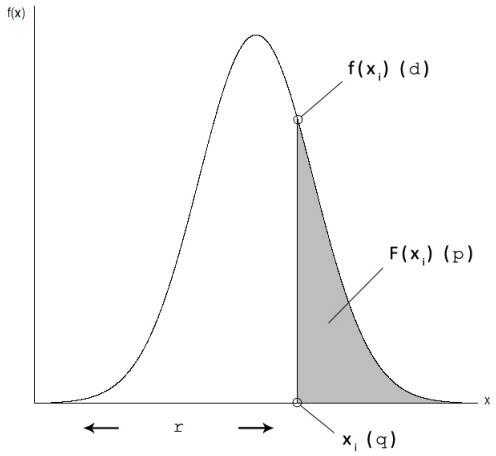


28. Les lois de probabilité continues – généralités

Ces lois s'appliquent à des variables quantitatives continues.

Paramètres :
- x_i : chaque valeur possible de la variable continue x. Également appelée quantile
- $f(x_i)$: distribution de probabilité de la valeur x_i. Également appelée densité de probabilité de x_i. Comprise entre 0 et 1
- $F(x_i)$: aire sous la courbe située à droite ou à gauche de x_i, suivant la situation. Également appelée fonction de répartition de x_i. On note $F(x_i)_{\text{droite}} = P(x > x_i)$ et $F(x_i)_{\text{gauche}} = P(x \leq x_i)$. Comprise entre 0 et 1.

Dans **R** :
- `dY()` : donne la densité de probabilité $f(x_i)$ pour une distribution de type `Y`
- `pY()` : donne la fonction de répartition $F(x_i)$ pour une distribution de type `Y`. **R** considère par défaut la répartition à gauche, préciser `lower.tail=FALSE` pour la répartition à droite
- `qY()` : donne la valeur x_i de la variable x correspondant à une valeur de $F(x_i)$ pour une distribution de type `Y`. **R** considère par défaut la répartition à gauche de k, préciser `lower.tail=FALSE` pour la répartition à droite
- `rY()` : donne une série de valeurs aléatoires de la variable x pour une distribution de type `Y`.

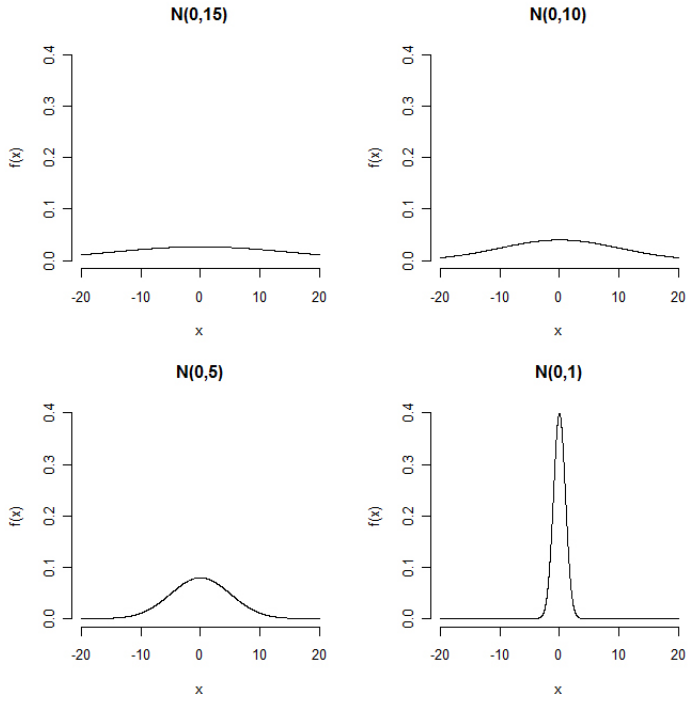


29. La loi normale

Écriture : $N(\mu, \sigma)$
avec :
μ : moyenne de la variable x
σ : écart-type de la variable x

Cas particulier, la loi normale centrée-réduite : $N(0,1)$

Dans **R** (voir fiche **28** pour des explications) :
`dnorm(xi,mu,sigma)`
`pnorm(xi,mu,sigma)`
`qnorm(F(xi),mu,sigma)`
`rnorm(z,mu,sigma)` avec `z` : nombre de valeurs à générer

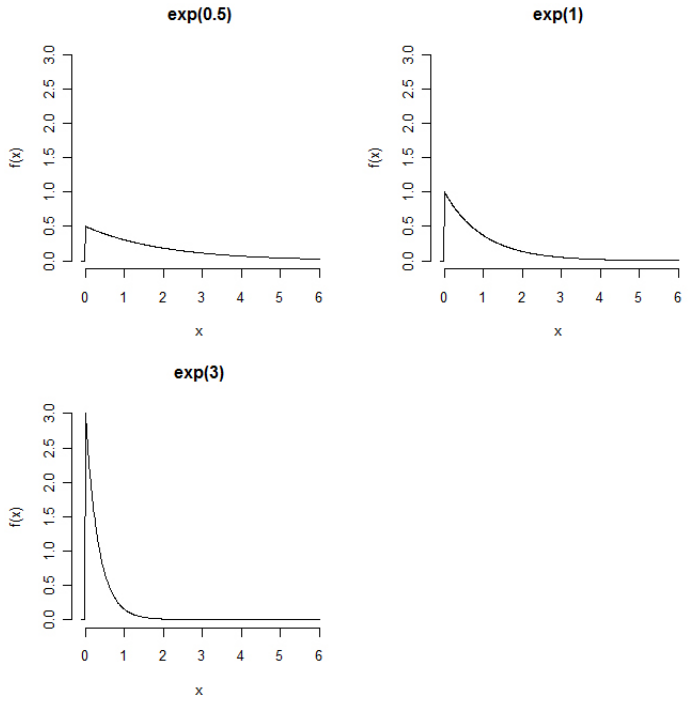


30. La loi exponentielle

La loi exponentielle correspond souvent à des évènements dont la probabilité de survenue diminue avec le temps. Elle est également utilisée pour modéliser des durées de vie.

Écriture : $exp(\lambda)$
avec λ : paramètre de la loi ($0 < \lambda < +\infty$)

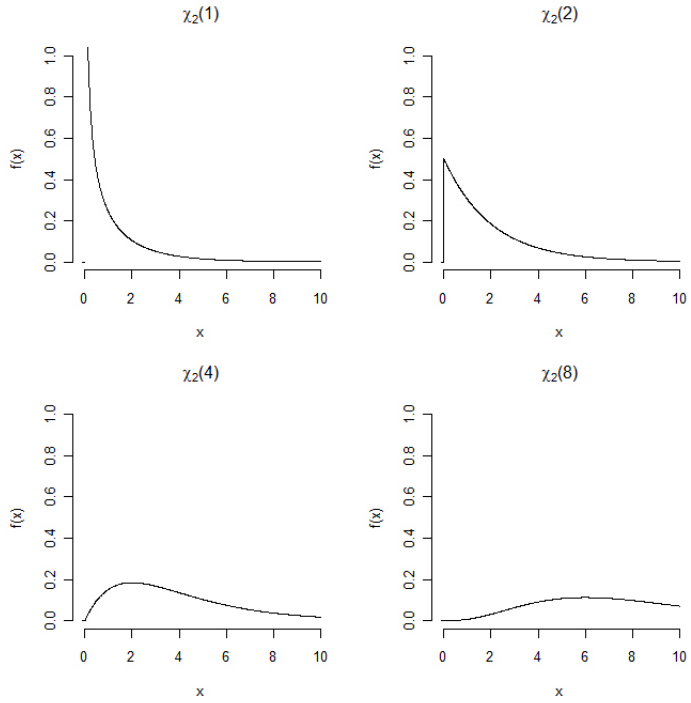
Dans R (voir fiche **28** pour des explications) :
`dexp(xi,lambda)`
`pexp(xi,lambda)`
`qexp(F(xi),lambda)`
`rexp(z,lambda)` avec `z` : nombre de valeurs à générer



31. La loi de χ^2

Écriture : $\chi^2(\nu)$
avec ν : nombre de degrés de liberté (ddl), *i.e.* de paramètres indépendants impliqués dans la loi $(0 < \nu < +\infty)$

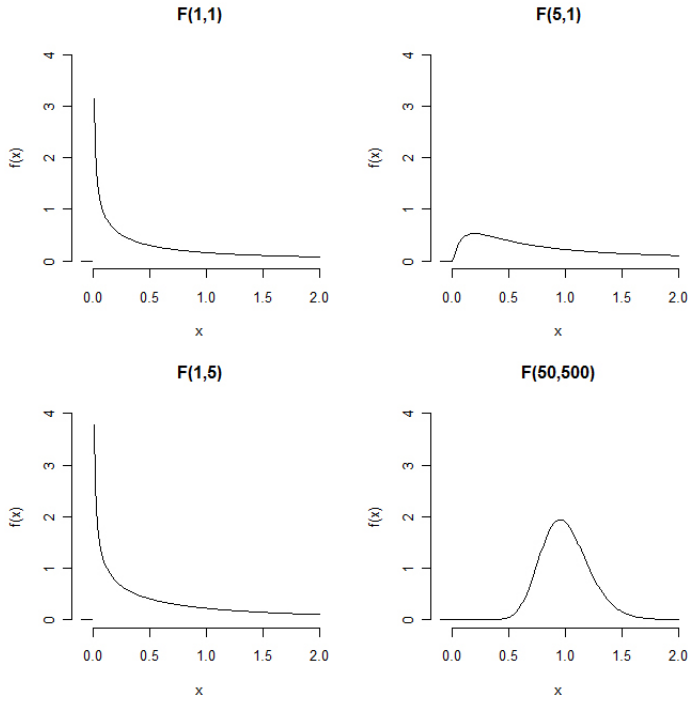
Dans **R** (voir fiche **28** pour des explications) :
`dchisq(xi,ddl)`
`pchisq(xi,ddl)`
`qchisq(F(xi),ddl)`
`rchisq(z,ddl)` avec `z` : nombre de valeurs à générer



32. La loi de Fisher - Snedecor

Écriture : $F(\nu_1, \nu_2)$
avec :
ν_1 : 1er nombre de degrés de liberté (ddl) $(0 < \nu_1 < +\infty)$
ν_2 : 2ème nombre de ddl $(0 < \nu_2 < +\infty)$

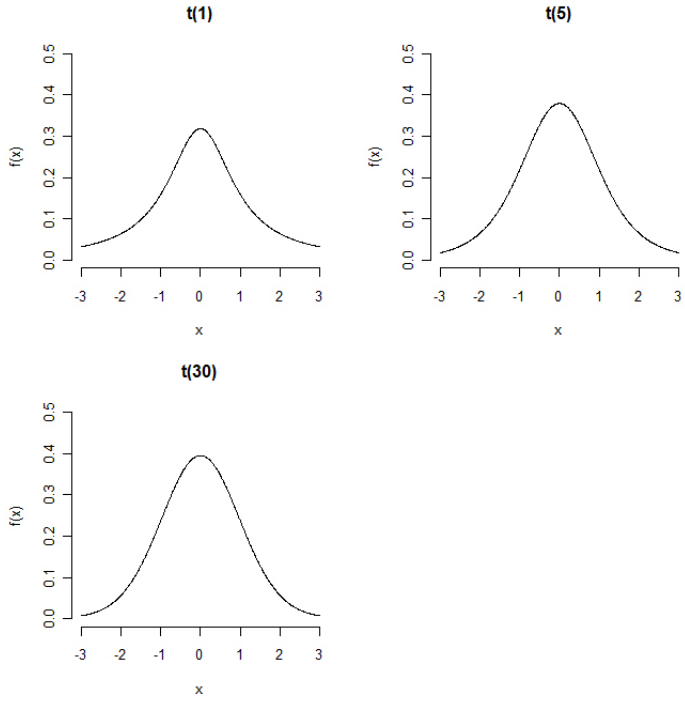
Dans R (voir fiche **28** pour des explications) :
`df(xi,ddl1,ddl2)`
`pf(xi,ddl1,ddl2)`
`qf(F(xi),ddl1,ddl2)`
`rf(z,ddl1,ddl2)` avec `z` : nombre de valeurs à générer



33. La loi de Student

Écriture : $t(\nu)$
avec ν : nombre de degrés de liberté (ddl) $(0 < \nu < +\infty)$

Dans R (voir fiche **28** pour des explications) :
`dt(xi,ddl)`
`pt(xi,ddl)`
`qt(F(xi),ddl)`
`rt(z,ddl)` avec `z` : nombre de valeurs à générer



34. Principe des tests statistiques et risques associés à la conclusion

Le principe de réalisation de tout test statistique est le suivant :

1. on pose une hypothèse nulle H_0, de type « rien à signaler » (ex : les moyennes μ_A et μ_B sont égales) ou « valeur ponctuelle » (ex : la moyenne $\mu = 10$, la proportion $p = 50$ %)
2. on pose une hypothèse H_1, de telle manière que H_0 et H_1 soient exclusives (ex : les moyennes μ_A et μ_B sont différentes, la moyenne $\mu \neq 10$)
3. on calcule la valeur de la Variable de Test (VT), d'après une formule qui dépend du test utilisé
4. on utilise la valeur de la VT calculée pour déterminer une *p-value, i.e.* une probabilité d'obtenir la valeur mesurée (moyenne, pourcentage…) si H_0 est vraie
5. on conclut sur les deux hypothèses posées grâce à cette *p-value* :
   - si la *p-value* est supérieure au seuil α fixé avant le test (5 % en général, voir fiche **35**), ne pas rejeter H_0
   - si la *p-value* est inférieure au seuil α, rejeter H_0.

Conclure sur les deux hypothèses présente deux risques :
- le risque de 1ère espèce ou risque α, qui correspond au risque de rejeter de H_0 si celle-ci est réellement vraie (voir fiche **35**)
- le risque de 2ème espèce ou risque β, qui correspond au risque de ne pas rejeter de H_0 si celle-ci est réellement fausse (voir fiche **37**).

	Réalité (inconnue le plus souvent)	
Décision	H_0 vraie	H_0 fausse
H_0 non rejetée	Bonne décision	Erreur β
H_0 rejetée	Erreur α	Bonne décision

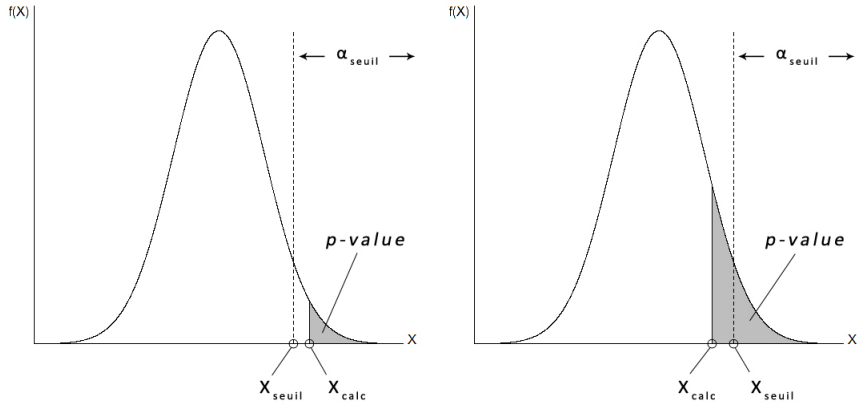
La probabilité associée au fait de rejeter H_0 si celle-ci est fausse (soit $1-\beta$) est appelée *puissance* du test. Cette notion est très importante car elle relativise le résultat des tests, et donc la conclusion que l'on en tire (voir fiche **37**).

35. Le risque ou seuil de rejet α

Le risque α, ou seuil de rejet ou encore seuil de signification de l'hypothèse H_0, est une valeur fixée arbitrairement *avant la réalisation de tout test statistique*. Elle correspond à un risque assumé de se tromper, celui de rejeter H_0 *si celle-ci est réellement vraie* (ce que bien sûr on ne sait jamais).

Lorsque l'on calcule une *p-value* suite à un test (voir fiche **34**), on la compare au seuil α choisi :

- si la *p-value* est inférieure au seuil α, on rejette H_0. Il faut être conscient que l'on prend un risque α de se tromper
- si la *p-value* est supérieure au seuil α, on ne rejette pas H_0. Il est alors intéressant de calculer la puissance du test que l'on vient d'utiliser (voir fiches **37** et **38**). Obtenir une *p-value* supérieure au seuil α peut en effet avoir deux origines :
  1. une réelle véracité de H_0
  2. un manque de puissance du test, *i.e.* un risque (β, voir fiche **34**) important de « passer à côté » d'une réelle fausseté de H_0 (voir fiche **37**).



X_{seuil} : valeur de la Variable de Test (VT) X qui donne une fonction de répartition à droite égale au seuil α (test unilatéral droit ici).

X_{calc} : valeur de la VT X calculée à partir de l'échantillon testé.

À gauche l'hypothèse H_0 est rejetée, à droite elle ne l'est pas.

D'un point de vue pratique, on utilise souvent un seuil α de 5 %. Cela veut dire qu'on assume un risque de 5 % de rejeter H_0 si celle-ci est réellement vraie.

36. La correction du seuil de rejet α (ou des *p-values*)

Si une série de tests statistiques est réalisée avec à chaque fois α pour seuil de rejet de H_0 (voir fiche **35**), le risque global de rejeter H_0 si celle-ci est vraie augmente. En effet, plus on effectue de tests, plus on a de chance de tomber sur un échantillon peu représentatif de la population dont il provient (donnant une *p-value* inférieure au seuil α).

Il est donc nécessaire de corriger le seuil de rejet α de chaque test lorsque plusieurs sont réalisés (ou leur *p-value*, ce qui revient au même), afin que le risque global soit égal au α souhaité.

Cette situation se présente :

- lorsque les tests vont permettre de prendre une décision unique, dès que l'un d'eux au moins permet le rejet de H_0
- lorsqu'est réalisée une série de tests deux-à-deux, soit directement soit après une analyse globale (ANOVA, test du χ^2 d'homogénéité…).

Plusieurs méthodes de correction existent, dont les trois suivantes.

La technique de Bonferroni

Si k tests sont effectués, la technique consiste simplement à diviser le seuil de rejet global α par k, donc considérer pour chaque test le seuil de rejet $\frac{\alpha}{k}$.

Cela revient à multiplier chaque *p-value* par k, sans changer le seuil α.

La technique séquentielle de Holm

La procédure se réalise en plusieurs étapes :

1. classer les *p-values* de tous les tests réalisés par ordre croissant ($p_1 < \ldots < p_k$), k étant le nombre de tests effectués
2. rejeter H_0 pour les tests dont la *p-value* satisfait la condition :

$$p_i \leq \frac{\alpha_{\text{seuil}}}{k+1-i}$$

où i est le rang de la *p-value* après classement.

La technique du False Discovery Rate (FDR) de Benjamini et Hochberg

La procédure se réalise en plusieurs étapes :

1. classer les *p-values* de tous les tests réalisés par ordre croissant ($p_1 < \ldots < p_k$), k étant le nombre de tests effectués
2. rejeter H_0 pour les tests dont la *p-value* satisfait la condition :

$$p_i \leq \alpha_{\text{seuil}} \times \frac{i}{k}$$

où i est le rang de la *p-value* après classement.

La technique la plus stricte est celle de Bonferroni, la moins stricte celle du FDR. Cette dernière peut être appliquée par défaut. Dans tous les cas la méthode de correction du seuil de rejet de H_0 doit être décidée *avant de réaliser les tests.*

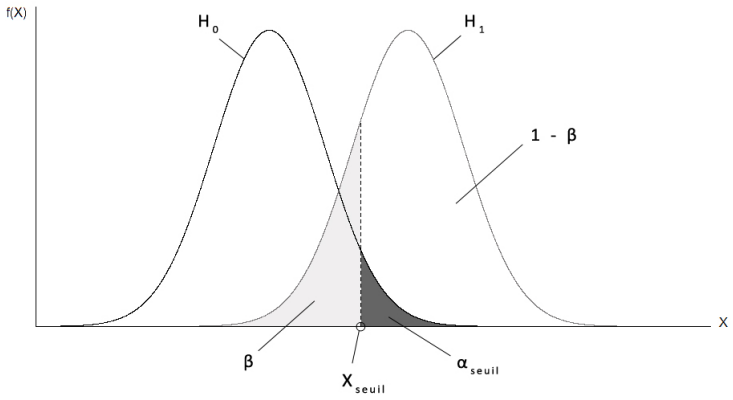
Dans **R**, si `p` est un vecteur contenant les *p-values* non corrigées, utiliser la fonction `p.adjust()` pour récupérer un vecteur avec les *p-values* corrigées (dans le même ordre) :
`p.adjust(p,method="bonferroni")` pour la correction de Bonferroni
`p.adjust(p,method="holm")` pour la correction de Holm
`p.adjust(p,method="BH")` ou `p.adjust(p,method="fdr")` pour la correction de Benjamini et Hochberg (FDR).

37. Le risque β et la puissance du test

Le risque β est le risque de ne pas rejeter l'hypothèse H_0 *si celle-ci est réellement fausse*. Contrairement au risque α, le risque β ne peut pas être fixé. En effet, si α dépend de la distribution de la Variable de Test (VT) sous H_0 (voir fiche **35**), β dépend de sa distribution sous H_1. Or cette distribution est inconnue, puisque l'hypothèse H_1 regroupe une infinité de distributions (ex : si l'hypothèse H_1 est $\mu_A \neq \mu_B$, les deux moyennes μ_A et μ_B peuvent différer d'une infinité de façons).

La puissance d'un test représente la probabilité de rejeter H_0 *si celle-ci est réellement fausse* (*i.e.* de faire le bon choix). Elle équivaut à $1-\beta$, et est donc également une variable dépendant de la distribution de la VT sous H_1.

Le risque β et la puissance du test dépendent du seuil α fixé :



X_{seuil} : valeur de la VT X qui donne une fonction de répartition à droite égale au seuil α pour la distribution sous H_0 (test unilatéral droit ici).

La puissance d'un test augmente :
- quand augmente le seuil α
- quand augmente l'effectif de l'échantillon testé (ce qui diminue l'étalement de la distribution de la VT ou éloigne les distributions de la VT sous H_0 et H_1, selon le test)
- quand augmente l'écart réel entre les valeurs testées (moyennes, proportions...).

Il est possible de calculer la puissance d'un test après avoir réalisé celui-ci (voir fiche **38**). Cela est indispensable car, lorsque la *p-value* d'un test est supérieure au seuil α fixé, cela peut aussi bien provenir d'une réelle véracité de H_0 que d'un manque de puissance du test alors que H_0 est réellement fausse. Avant de conclure sur H_0, il faut donc avoir une idée de la puissance du test employé (en particulier lorsque la *p-value* est peu supérieure au seuil α).

38. Calculer la puissance d'un test *a posteriori*

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]pwr

Il existe un lien entre le seuil de rejet α du test statistique utilisé (voir fiches **34** et **35**), la puissance de ce test (voir fiche **37**), la différence entre les échantillons pour la variable mesurée et la taille des échantillons. On peut donc déterminer – au moins approximativement – la puissance d'un test après l'avoir réalisé, car on connaît tous les autres paramètres.

Selon l'objectif de l'étude, la puissance à atteindre n'est pas la même. On considère en effet que lorsque l'objectif est de *rejeter* H_0 (voir fiche **34**), la puissance du test doit être d'au moins 80 % ; lorsque l'objectif est de *ne pas rejeter* H_0 (voir fiche **34**), la puissance doit être d'au moins 95 %.

Dans R, la fonctions `power()` permet, pour quelques tests paramétriques courants, de calculer leur puissance lorsque l'on connaît tous les autres paramètres. Lorsque l'on a choisi ou été contraint d'utiliser un test non paramétrique, on peut tout de même estimer sa puissance. Celle-ci est vaut en effet environ 95 % de la puissance du test paramétrique équivalent.

Toutes les fonctions décrites sont basées sur le même principe : le paramètre `power` doit avoir comme valeur `NULL` tandis que tous les autres doivent être fixés. Toutes les fonctions considèrent que l'effectif est le même dans les différents groupes à comparer (on parle d'effectifs *équilibrés*). Il est fortement conseillé de toujours prévoir ses plans d'échantillonnage ou d'expérience de cette façon, ce qui (i) facilite l'analyse et (ii) permet d'utiliser la plupart des tests non paramétriques, dont l'équilibre des effectifs est une condition. Cela dit, si les effectifs ne sont pas équilibrés, on peut tout de même se faire une idée de la puissance du test employé en utilisant comme effectif celui du *plus petit* des groupes comparés. Mieux vaut en effet légèrement sous-estimer la puissance d'un test pour conclure, que la sur-estimer.

Comparaison d'une moyenne avec une valeur théorique (voir fiche 63) ou de deux moyennes (voir fiche 64) – test *t* de Student

```
power.t.test(n,delta,sd,sig.level,power,type)
```

avec :
`n` : effectif de chaque groupe
`delta` : différence entre la moyenne de l'échantillon et la moyenne théorique, ou entre les deux moyennes
`sd` : écart-type (identique pour les deux échantillons dans le cas d'une comparaison de deux moyennes), dans l'unité des moyennes
`sig.level` : seuil de rejet α utilisé pour le test
`power` : puissance du test (`NULL`)
`type` : type de test (`"one.sample"` pour la comparaison d'une moyenne avec une valeur théorique, `"two.sample"` pour la comparaison de deux moyennes, `"paired"` pour la comparaison de deux moyennes en séries appariées).

Comparaison de plus de deux moyennes – analyse de variance à un facteur (voir fiche 67)

```
power.anova.test(groups,n,between.var,within.var,sig.level,
power)
```

avec :
`groups` : nombre de modalités à comparer
`between.var` : somme des carrés des écarts intergroupe
`within.var` : somme des carrés des écarts intragroupe (identique pour toutes les modalités).

Comparaison de deux proportions – test du χ^2 d'homogénéité (voir fiche 60)

```
power.prop.test(n,p1,p2,sig.level,power)
```

avec `p1`, `p2` : proportions observées.

Significativité d'un coefficient de corrélation linéaire de Pearson (voir fiche 76)

`pwr.r.test(n,r,sig.level,power)`[1]

avec `r` : coefficient de corrélation observé.

39. L'identification et la suppression des données aberrantes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`outliers`

L'identification des données aberrantes est une étape obligatoire de toute analyse statistique. Elle doit se faire avant tout *visuellement*, grâce à des graphes du type histogramme (voir fiche **12**) ou boîtes à moustaches (voir fiche **13**).

On peut également s'aider du test de Grubbs pour repérer des données sinon aberrantes, du moins très différentes des autres. Utiliser pour cela `grubbs.test(serie)`[1] où `serie` est un vecteur contenant la série de données. La fonction teste la valeur extrême de la série la plus éloignée de la moyenne. Pour tester l'autre valeur extrême, ajouter l'argument `opposite=TRUE`.

La suppression d'éventuelles données aberrantes est un point hautement plus délicat que leur simple identification. Supprimer une ou plusieurs donnée(s) aura nécessairement un effet sur les analyses qui vont suivre, et cet effet sera d'autant plus important que l'effectif de l'échantillon est faible. Il est donc tentant de supprimer les individus qui orientent les résultats vers des conclusions inverses à celles qui sont attendues.

Il n'y a globalement que deux raisons qui doivent pousser à éliminer une donnée :

- s'il y a manifestement eu une erreur technique dans la mesure ou dans la retranscription de la donnée (par exemple si l'appareil de mesure est défectueux)
- si la valeur de la mesure obtenue est biologiquement improbable pour la variable mesurée.

En dehors de ces deux cas, il y a de grandes chances pour que la valeur « aberrante » soit simplement une singularité biologique de l'individu mesuré. Cette singularité est l'expression de la variabilité naturelle de tout caractère biologique, rien ne justifie donc que l'individu soit supprimé de l'étude.

On notera donc en particulier qu'un résultat significatif du test de Grubbs (ou de tout autre test équivalent) ne doit pas nécessairement conduire à l'élimination de l'individu repéré comme « aberrant », mais simplement à s'intéresser de plus près à lui.

Dans tous les cas, l'identification et la suppression éventuelle de données aberrantes doit se faire *avant tout autre analyse*.

40. L'intervalle de confiance et l'erreur standard

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire, [2]boot

Calculer l'intervalle de confiance d'une moyenne

Petit effectif (< 30 individus) : utiliser le module « intervalle de confiance » du test de Mann - Whitney - Wilcoxon (test non paramétrique qui calcule en fait l'intervalle de confiance de la *médiane*) : `wilcox.test(serie,conf.int=TRUE)$conf.int` où `serie` est un vecteur contenant la série de données. Si les conditions du test de Mann - Whitney - Wilcoxon sont réunies (voir fiche **64**), médiane et moyenne sont proches.

Grand effectif (> 30 individus) : utiliser le module « intervalle de confiance » du test *t* de Student (test paramétrique) : `t.test(serie)$conf.int`.

Calculer l'erreur standard d'une moyenne

Quel que soit l'effectif : `se(serie)`[1].

Calculer l'intervalle de confiance d'une proportion

Quel que soit l'effectif, utiliser le module « intervalle de confiance » du test binomial exact : `binom.test(a,b)$conf.int` où `a` est le nombre d'individus de la catégorie d'intérêt et `b` l'effectif total.

— EXEMPLE(S) —

L'intervalle de confiance d'un sex-ratio de 9 femelles sur 25 individus est :

```
> binom.test(9,25)$conf.int
[1] 0.1797168 0.5747937
```

Calculer l'intervalle de confiance de n'importe quoi

La technique utilisée est celle du bootstrap non paramétrique : `bootstrap(serie,function(x,i) mean(x[i]))`[1,2]. Dans cet exemple une moyenne est calculée, mais la fonction gère des expressions bien plus complexes.

La syntaxe particulière de cette fonction doit obéir à deux règles :

- l'argument utilisé dans l'expression à calculer (ici cette expression est `mean(x[i])`) doit être le même que le 1er déclaré à `function()`. Ici cet argument est `x`
- l'argument utilisé dans l'expression à calculer doit toujours être suivi du second déclaré à `function()` placé entre crochets. Celui-ci est par défaut `i`, voir l'aide de la fonction `boot()`[2] pour plus d'informations.

— EXEMPLE(S) —

```
> variable <- c(1:10)
> bootstrap(variable,function(x,i) mean(x[i]))[1,2]
[...]
95 percent confidence interval:
 3.7 7.3
sample estimates:
original value
           5.5
```

Pour toutes ces fonctions, la précision de l'intervalle de confiance peut être modifiée grâce à l'argument `conf.level` (par défaut `conf.level=0.95`, ce qui calcule l'intervalle de confiance à 95 %).

41. Tracer un diagramme en barres avec barres d'erreur

L'exemple présenté ici traite de moyennes et de barres d'erreur représentant des erreurs standards. Il peut bien sûr être adapté à n'importe quelles valeurs.

Un facteur

L'étape préliminaire est de rassembler les moyennes (une par modalité du facteur) dans un vecteur `moyennes` (contenant les valeurs dans l'ordre du graphe, de gauche à droite) et les erreurs standards (voir fiche **40**) dans un vecteur `erreurs` (avec les valeurs dans le même ordre que les moyennes). La fonction `tapply()` est très utile dans cette situation (voir fiche **14**).
La procédure est ensuite la suivante :

```
> abscisses <- barplot(moyennes)
> segments(abscisses,moyennes-erreurs,abscisses,moyennes+erreurs)
> segments(abscisses-0.15,moyennes-erreurs,abscisses+0.15,moyennes-erreurs)
> segments(abscisses-0.15,moyennes+erreurs,abscisses+0.15,moyennes+erreurs)
```

Deux facteurs

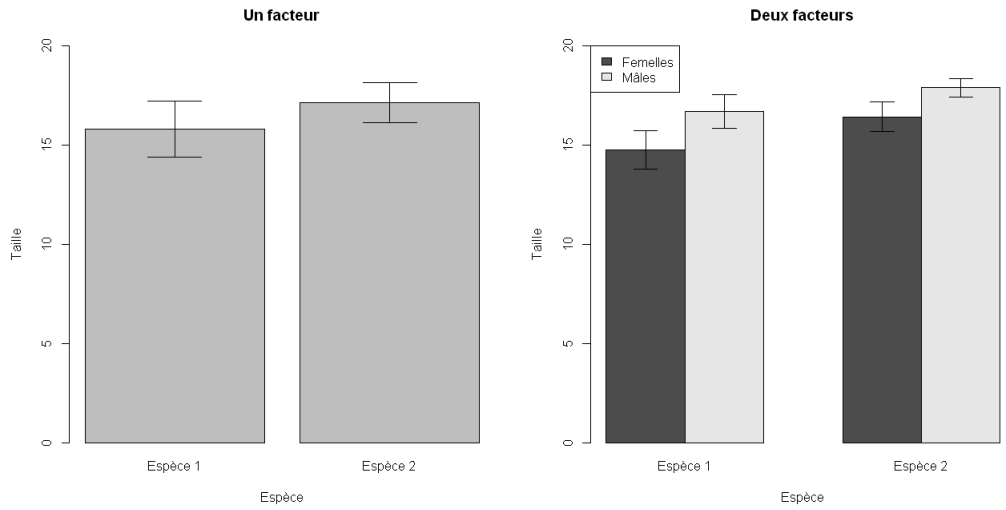
Moyennes et erreurs standards doivent être contenues dans des matrices. Ces matrices doivent avoir en lignes les modalités du 2nd facteur et en colonnes celles du 1er facteur (la fonction `tapply()` est là encore très utile, voir fiche **14**). La procédure est ensuite identique, il faut seulement ajouter l'argument `beside=TRUE` à la fonction `barplot()`.

L'argument `names.arg=noms` de la fonction `barplot()` ajoute ou modifie le nom des barres (`noms` étant un vecteur contenant les noms de gauche à droite), tandis que `legend=TRUE` ajoute une légende dans le cas de deux facteurs.

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.

Pour modifier la légende de l'axe vertical, utiliser l'argument `ylab="Légende"`.

Pour (beaucoup) plus d'options graphiques, voir `?barplot` (options spécifiques aux diagrammes en barre), `?legend` (options spécifiques de la légende) et `?par` (options graphiques générales).



42. Les différents types de test statistique

Il existe quatre grandes familles de tests statistiques, qui se différencient par leur objectif :
- les tests d'homogénéité : ils permettent de tester l'égalité d'un paramètre entre plusieurs populations (ex : égalité des moyennes entre n populations)
- les tests de conformité : ils permettent de tester l'égalité d'un paramètres d'une (ou plusieurs) population(s) à une (ou plusieurs) valeur(s) théorique(s) (ex : proportion égale à 50 %)
- les tests d'indépendance : ils permettent de tester si deux variables mesurées sont indépendantes l'une par rapport à l'autre
- les tests d'ajustement : ils permettent de tester si une distribution de valeurs observée est conforme à une distribution théorique ou si deux distributions observées sont identiques.

Ces quatre catégories ne sont pas exclusives, un même test pouvant avoir plusieurs objectifs (ex : le test du χ^2 de conformité est aussi un test d'ajustement).

Le choix d'un test statistique repose sur plusieurs critères :
- son objectif : à quelle question doit-il permettre de répondre ?
- le type de données :
  - comment ont-elles été récoltées (voir fiches **6** et **7**) ?
  - combien de variables ont été mesurées ?
  - quelle est la nature de ces variables (voir fiche **5**) ?
  - comment ces variables sont-elles distribuées ?
- la quantité de données : combien d'individus ont été comptés/mesurés ?
- le nombre de populations à comparer.

La réponse à toutes ces questions peut généralement être apportée avant même de récolter les données, dès la préparation de l'étude. Une bonne démarche consiste ainsi à choisir dès le départ les tests qui vont être utilisés. Ce choix ne pose pas de problèmes si les questions auxquelles l'étude doit répondre sont clairement identifiées.

43. Le caractère aléatoire et simple d'une série de données

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lawstat`

Cette condition est l'une des plus fondamentales d'un grand nombre de tests statistiques, et en particulier de *tous ceux présentés dans cet ouvrage.* On peut la remplir (et il est *très fortement* conseillé de le faire) dès la mise en place du plan d'échantillonnage (voir fiche **6**) ou du plan d'expérience (voir fiche **7**).

On peut également tester le caractère aléatoire et simple d'une série de données grâce au test non paramétrique de Bartels : `bartels.test(serie)`[1] où `serie` est un vecteur contenant des données numériques ou non (ex : modalités d'un facteur), dans l'ordre où elles ont été récoltées.

Une *p-value* significative indique qu'il existe une corrélation entre les valeurs de la série, négative ou positive (par défaut le test est bilatéral). Pour connaître le signe de cette corrélation, réaliser un test unilatéral en ajoutant l'argument `alternative="positive.correlated"` ou `alternative="negative.correlated"`. Dans ce cas l'hypothèse alternative H_1 (voir fiche **34**) n'est plus « il y a une corrélation entre les valeurs » mais « il y a une corrélation positive » ou « il y a une corrélation négative » entre les valeurs.

44. L'ajustement à une distribution théorique

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]car, [2]RVAideMemoire

Les tests d'ajustement à une distribution théorique peuvent être une fin en soi mais sont souvent préliminaires à d'autres tests. La condition qu'une distribution observée suive une loi normale est en particulier très fréquemment requise. Ces tests ne sont pas obligatoires (et parfois ne peuvent pas être utilisés). En effet, l'ajustement peut (et doit *avant tout*) être testé graphiquement.

Test graphique

Le test visuel est réalisé grâce au graphe quantile-quantile, tracé de la manière suivante : `qqPlot(serie,dist="loi",par)`[1] où `serie` est un vecteur contenant la série de données, `loi` est la loi théorique choisie (entre guillemets) et `par` ses paramètres séparés par une virgule (voir fiches **24** à **33**).

— EXEMPLE(S) —

Ajustement à une loi binomiale (voir fiche **25**) :

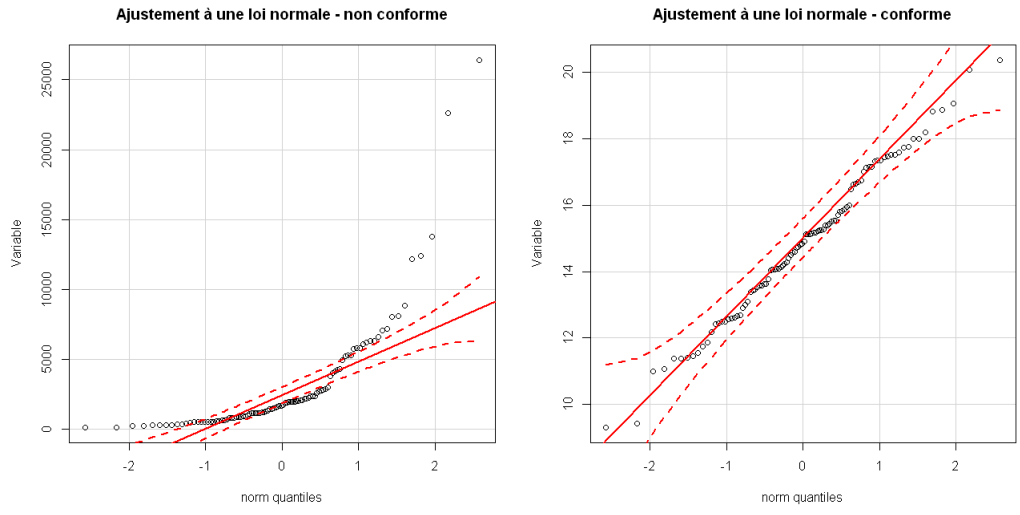
```
> qqPlot(serie,dist="binom",n,p)[1]
```

Ajustement à une loi exponentielle (voir fiche **30**) :

```
> qqPlot(serie,dist="exp",lambda)[1]
```

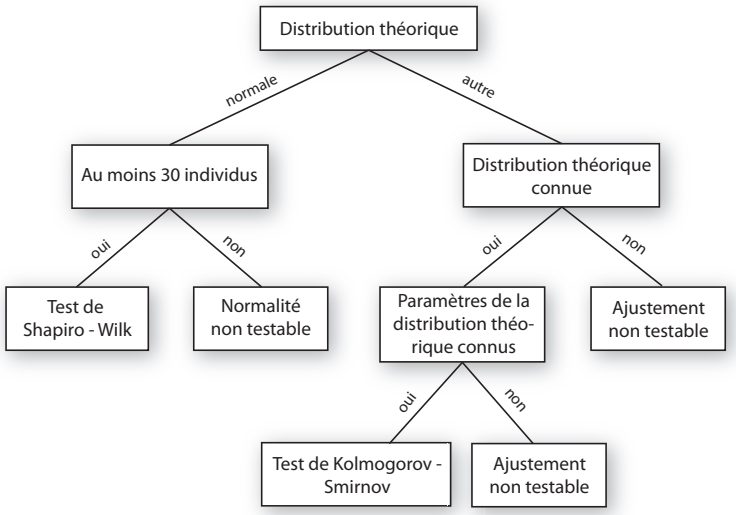
Dans le cas d'une loi normale, le même résultat est obtenu directement *via* `qqPlot(serie)`[1], la loi normale étant celle utilisée par défaut. Pour tracer un graphe par niveau d'un facteur : `byf.qqnorm(variable~facteur)`[2] où `variable` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ». Dans le cas d'un ajustement à une loi normale multivariée, utiliser les fonctions `mqqnorm()`[2] et `byf.mqqnorm()`[2] (voir `?mqqnorm` et `?byf.mqqnorm` pour leur utilisation).

La distribution de la série suit la loi théorique choisie si les points du graphe sont à peu près alignés sur une droite. Toute autre structuration des points (courbure(s), nombreux points éloignés...) indique le contraire. La droite tracée est celle qui passe par les 1er et 3ème quartiles de la distribution représentée, et son intervalle de confiance est affiché en pointillés. Les points peuvent ne pas être parfaitement alignés sur la droite, mais s'ils restent à peu près dans l'intervalle de confiance l'ajustement est considéré comme correct.



Test statistique

Les tests suivants exigent que la variable testée soit *quantitative et continue*. Pour choisir le test approprié :



Ajustement à une loi normale – Test de Shapiro - Wilk

Pour réaliser le test : `shapiro.test(serie)`. Pour tester la normalité de la distribution d'une variable par niveau d'un facteur : `byf.shapiro(variable~facteur)`[2] où `variable` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre). Dans le cas d'un ajustement à une loi normale multivariée, utiliser les fonctions `mshapiro.test()`[2] et `byf.mshapiro()`[2] (voir `?mshapiro.test` et `?byf.mshapiro` pour leur utilisation).

Ajustement à une loi autre que normale – Test de Kolmogorov - Smirnov

Pour réaliser le test : `ks.test(serie,ploi,par)` où `loi` est la loi choisie et `par` ses paramètres séparés par une virgule (voir fiches **24** à **33**).

EXEMPLE(S)

Ajustement à une loi de χ^2 (voir fiche **31**) :

```
> ks.test(serie,pchisq,ddl)
```

Ajustement à une loi de Fisher - Snedecor (voir fiche **32**) :

```
> ks.tset(serie,pf,ddl1,ddl2)
```

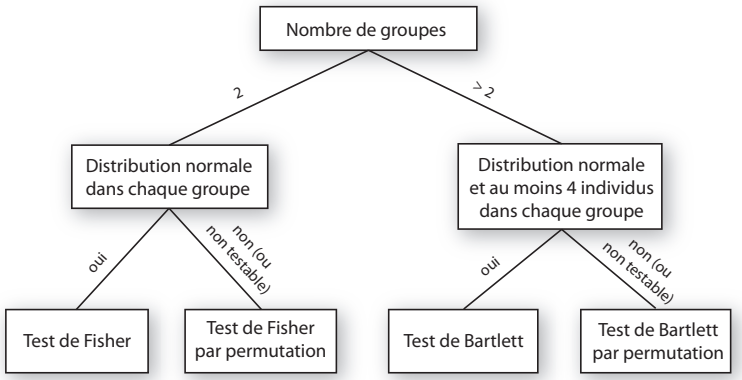
45. L'homogénéité des variances de plusieurs séries de données

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Les tests d'homogénéité de variances peuvent être une fin en soi mais sont souvent préliminaires à d'autres tests. L'homogénéité des variances (ou *homoscédasticité*) doit en effet au minimum être contrôlée, au maximum assurée dans de nombreux tests usuels.

Pour choisir le test approprié :



Comparaison des variances de deux populations

Test de Fisher (paramétrique)

Pour réaliser le test : `var.test(variable~facteur)` où `variable` est un vecteur contenant les valeurs de la variable mesurée et `facteur` un vecteur contenant la classe de chaque individu (dans le même ordre que `variable`). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Test de Fisher par permutation (non paramétrique)

Pour réaliser le test : `perm.var.test(variable~facteur)`[1].

Comparaison des variances de plus de deux populations

Test de Bartlett (paramétrique)

Pour réaliser le test : `bartlett.test(variable~facteur)`.

Test de Bartlett par permutation (non paramétrique)

Pour réaliser le test : `perm.bartlett.test(variable~facteur)`[1].

46. Les transformations de variable

Les tests paramétriques sont en général plus puissants que leurs homologues non paramétriques. Cependant ils ont des conditions d'application plus rigoureuses (notamment normalité des distributions et homoscédasticité). Lorsque celles-ci ne sont pas remplies, il y a deux possibilités : (i) utiliser un test homologue non paramétrique moins puissant ou (ii) transformer la variable pour satisfaire la ou les conditions non remplie(s) et utiliser le test paramétrique. La conclusion d'un test sur des données transformées peut en effet s'appliquer, globalement, aux données initiales.

Les transformations usuelles sont les suivantes :

- logarithme : `variable2<-log10(variable)`. À utiliser lorsque la moyenne des échantillons (si plusieurs sont disponibles) est proportionnelle à leur écart-type. En histogramme cela donne un fort biais à gauche de la distribution. Cette transformation est fréquemment utilisée lorsque des processus de croissance ou de multiplication sont en jeu : masse, rendement, décompte d'organismes vivants…
- racine carrée : `variable2<-sqrt(variable)`. À utiliser lorsque la moyenne des échantillons est proportionnelle à leur variance (ce qui est exactement le cas pour les distributions de Poisson). En histogramme cela donne un léger biais à gauche
- inverse : `variable2<-1/variable`. À utiliser lorsque la variable est distribuée « en i », *i.e.* lorsque les valeurs les plus faibles sont les plus fréquentes.

Il existe un très grand nombre de transformations. À mesure que les techniques statistiques se développent, ces transformations deviennent toutefois de moins en moins nécessaires. Ainsi, les transformations « logarithme népérien + 1 » et « arc sinus racine carrée », qui servent à normaliser respectivement des données d'abondance et des proportions pour utiliser ensuite une analyse de variance, ne sont plus justifiées car les modèles linéaires généralisés (GLMs) sont adaptés à ce type de données (voir les fiches sur les comparaisons d'effectifs et de proportions).

Dans tous les cas, chercher une transformation complexe qui permettra de satisfaire une condition exigée par un test n'est pas conseillé. Si une transformation simple ne fonctionne pas, c'est que les données sont vraiment « particulières ». Il convient alors (i) de chercher d'où vient cette particularité, ce qui peut être biologiquement intéressant et (ii) d'utiliser un test non paramétrique.

Une fois qu'une transformation a été appliquée, il est nécessaire de repasser par un examen graphique des données (voir fiches **12**, **13** et **15**) et par le(s) test(s) qui avai(en)t conduit à les transformer (le plus souvent ajustement à une distribution théorique et/ou homoscédasticité, voir fiches **44** et **45**).

47. La démarche d'utilisation des modèles

L'utilisation des modèles en statistique (en tout cas en statistique appliquée à la biologie) suit une démarche qui est globalement toujours la même :

1. **Construction du modèle :** (i) on choisit le type de modèle que l'on utilise (linéaire, linéaire généralisé, non linéaire...) ; (ii) on choisit la loi de distribution sur laquelle le modèle doit être basé *a priori* ; (iii) on définit les relations entre les variables explicatives dans ce modèle. Dans le cadre de cet aide-mémoire, les étapes (i) et (ii) sont toujours proposées, tandis que l'étape (iii) doit nécessairement être réalisée par l'utilisateur puisqu'elle dépend de la question de recherche. Cette étape (iii) est basée sur la construction d'une *formule* (voir fiche **48**)
2. **Vérification de l'ajustement du modèle aux données :** il est important de faire cette vérification, car toute la suite de la démarche sera biaisée (voire complètement faussée) si le modèle n'est pas bien représentatif des données réelles. En pratique cette vérification passe par un examen des *résidus* du modèle (voir fiche **49**). Si le modèle n'est pas bien ajusté, il y a plusieurs possibilités : changer complètement de modèle, changer la loi sur laquelle il est basé, modifier les relations entre les variables explicatives, transformer les données (voir fiche **46**)... Dans tous les cas, tant que le modèle n'est pas bien ajusté aux données l'analyse ne doit pas aller plus loin.

La suite de la démarche peut prendre deux directions différentes, selon le type d'étude et le contexte :

- Approche par test de l'effet des variables explicatives :
  3. **Application d'un test sur le modèle :** l'effet de chaque terme de la formule est testé, ce qui conduit à l'obtention d'une *p-value* (voir fiche **50**)
  4. **Comparaisons multiples (si nécessaire) :** si un facteur à plus de deux niveaux a un effet significatif, réaliser des comparaisons multiples permet de conclure sur ceux qui diffèrent réellement (voir fiche **51**)
- Approche par sélection de modèle :
  3. **Sélection :** des modèles dérivés du premier mais n'incluant que certains termes sont comparés entre eux et avec le modèle complet. L'objectif est de sélectionner celui qui est le meilleur compromis entre un bon ajustement aux données et un petit nombre de variables explicatives (les deux variant en ses inverse ; voir fiche **52**).

48. La construction de la formule d'un modèle

La construction de formules est un élément essentiel de l'analyse statistique. Elle peut être relativement simple si les notions suivantes sont bien comprises :

- variable à expliquer et variable explicative
- variable quantitative et facteur (voir fiche **5**)
- facteur fixe et facteur aléatoire (voir fiche **5**)
- plan d'échantillonnage (voir fiche **6**) et plan d'expérience (voir fiche **7**).

Le principe de toute formule est le suivant : `variables.à.expliquer~variables.explicatives` (le symbole `~` signifie « expliqué par » ou « en fonction de »). En pratique il n'y a le plus souvent qu'une seule variable à expliquer, ce qui simplifie la partie gauche de la formule. C'est dans la partie droite, correspondant aux variables explicatives, qu'il est nécessaire de définir les relations entre ces variables. Il est important de comprendre que ces relations *dépendent de la question de recherche*. Les définir ne se fait donc pas au hasard, elles doivent au contraire représenter pertinemment la situation.

Il existe trois types de relation entre deux variables explicatives, représentées par trois symboles différents :

- l'effet de chaque variable est regardé *séparément* : `A + B`
- l'effet de chaque variable *ainsi que leur interaction* est regardé : `A * B`. Cette syntaxe est un raccourci strictement équivalent à `A + B + A:B`, qui signifie « variable A et variable B et l'interaction entre A et B ». Avec plus de variables, les choses se compliquent : `A * B * C` est équivalent à `A + B + C + A:B + A:C + B:C + A:B:C`, ce qui signifie « (i) les variables seules (A, B, C) et (ii) les interactions d'ordre 2 (entre A et B, entre A et C, entre B et C) et (iii) l'interaction d'ordre 3 (entre A, B et C). Il peut être tentant de mettre toutes les interactions possibles dans la formule, mais c'est une chose à éviter. En effet, à partir de l'ordre 3 les interactions sont très difficilement interprétables (ex : l'effet de A dépend de B, mais aussi de C...), sans compter que bien souvent les effectifs ne sont pas assez élevés pour évaluer des effets aussi fins. Il est toujours plus pertinent de se limiter aux effets que l'on peut expliquer
- un cas particulier, celui de deux facteurs *emboîtés* (ou *hiérarchisés*) : `A/B`, qui signifie « variable A et variable B emboîtée dans A » (ex : un facteur population emboîté dans un facteur région). On peut très bien étendre la relation à `A/B/C` (ex : si on considère des facteurs pays, région et population).

Ces trois symboles (quatre en réalité avec le `:`) permettent de définir toutes les relations entre les variables explicatives, qu'elles soient quantitatives ou qualitatives.

La gestion des facteurs aléatoires (voir fiche **5** à **7**) est un peu spéciale. Elle fait appel à une syntaxe particulière, qui dépend de la fonction utilisée. On rencontre deux cas :

- tests simples (non paramétriques généralement) : le facteur aléatoire est noté en fin de formule après le symbole `|`. Par exemple, dans la formule `reponse~facteur|bloc` le facteur `facteur` est fixe et le facteur `bloc` est aléatoire
- modèles mixtes : on appelle *mixte* un modèle qui contient au moins un facteur aléatoire. Plusieurs packages gèrent ces modèles (chacun avec une syntaxe différente), dans cet aide-mémoire c'est `lme4` qui est utilisé. Le facteur aléatoire est alors précisé, toujours en fin de formule, sous la forme `(1|facteur.alea)`. Par exemple, dans la formule `reponse~facteur+(1|bloc)` le facteur `facteur` est fixe et le facteur `bloc` est aléatoire. Plusieurs facteurs aléatoires peuvent être mis dans la formule, sous la forme `(1|facteur1) + (1|facteur2)` s'ils sont à effet indépendant ou `(1|facteur1/facteur2)` s'ils sont hiérarchisés. Il est possible de faire des choses bien plus complexes avec les modèles mixtes, voir **Bates** (2010) pour plus d'information.

Tous les fonctions créant un modèle, et plus généralement toutes les fonctions basées sur une formule, acceptent un argument `data`. Celui-ci permet de préciser le tableau de données dans lequel aller chercher les variables contenues dans la formule, ce qui évite d'allonger la syntaxe de la formule (et parfois de provoquer des erreurs).

49. La vérification de la validité d'un modèle

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

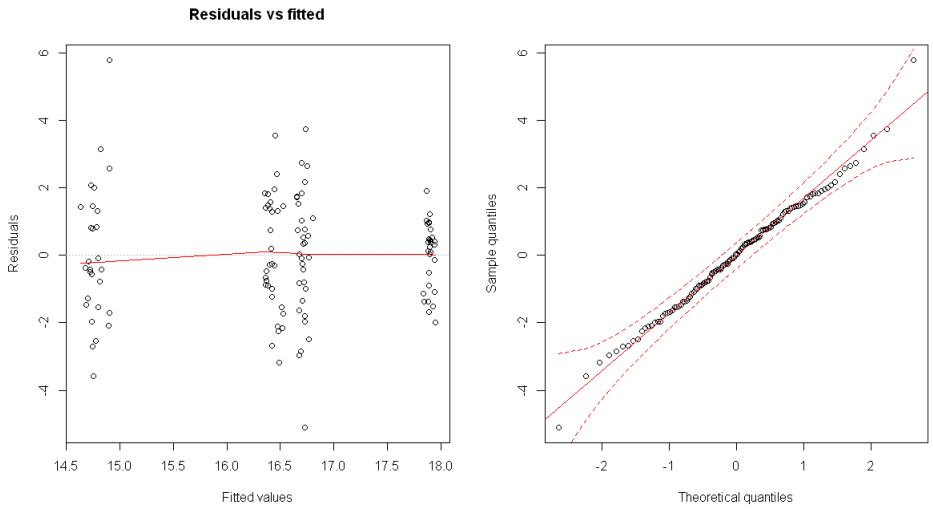
[1]RVAideMemoire

Écrire un modèle pour analyser des données est une chose, mais il est très important de vérifier que ce modèle s'ajuste bien aux données. Si ce n'est pas le cas, toute analyse découlant de ce modèle serait *a minima* biaisée, voire totalement invalide.

Les vérifications à effectuer pour valider un modèle se font essentiellement graphiquement et tournent globalement autour de trois points : l'*équivariance*, l'*indépendance* et la *normalité* des *résidus* du modèle (les résidus étant les écarts entre les valeurs réellement observées et celles prédites par le modèle, ces dernières étant nommées *fitted values*).

Ces trois vérifications essentielles sont réalisées grâce à la fonction `plotresid(modele)`[1]. Deux graphes sont renvoyés :

- le graphe de gauche sert à tester l'équivariance et l'indépendance des résidus. Sur ce graphe, la ligne rouge représente la tendance du nuage de points. Les hypothèses d'équivariance et d'indépendance sont acceptées lorsque cette ligne ne s'éloigne pas trop de l'horizontale. Plus précisément, l'hypothèse d'équivariance est acceptée lorsque la dispersion verticale des points est à peu près constante sur toute la longueur de l'axe des abscisses. L'hypothèse d'indépendance est acceptée lorsque l'orientation du nuage de points est horizontale
- le graphe de droite sert à tester la normalité des résidus. L'hypothèse de normalité est acceptée lorsque les points sont à peu près alignés sur une droite (voir fiche **44**).



En ajoutant l'argument `shapiro=TRUE` à la fonction, un test de Shapiro - Wilk est appliqué aux résidus (voir fiche **44**). Ce test n'est fiable que s'il y a au moins 30 individus dans les données du modèle.

50. L'application d'un test sur un modèle

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`car`

Après avoir vérifié que le modèle que l'on a construit est bien ajusté aux données (voir fiche **49**), l'effet des variables explicatives peut être testé. Il existe trois façons de réaliser ce test, que l'on appelle types I, II et III. S'il n'y a qu'une variable explicative, les trois types de test donnent exactement le même résultat. S'il y a plusieurs facteurs parmi les variables explicatives mais que les effectifs sont *équilibrés* (*i.e.* s'il y a autant d'individus dans toutes les classes d'un facteur et dans toutes les combinaisons des facteurs en interaction), il en est de même. Les trois types de test donnent des résultats différents s'il y a plusieurs facteurs et que les effectifs ne sont pas équilibrés. Il est important de comprendre d'où viennent ces différences, car elles reflètent des hypothèse statistiques (et donc biologiques) différentes. On prendra comme exemple un modèle du type `reponse~A+B+A:B`, où `A` et `B` sont des facteurs (voir fiche **48** pour une explication de la formule).

Type I : l'analyse est *séquentielle*, *i.e.* les termes sont testés dans l'ordre où ils apparaissent dans la formule (NB : lorsqu'un modèle est créé, **R** réorganise automatiquement la formule pour placer les facteurs seuls d'abord, puis les interactions d'ordre 2, les interactions d'ordre 3...). Dans notre exemple on a donc (i) un test de l'effet de `A` ; (ii) un test de l'effet de `B` *après avoir retiré* l'effet de `A` ; un test de l'effet de `A:B` après retiré les effets de `A` et `B`. Cela implique que les résultats du test pour les termes `A` et `B` sont *dépendants de l'ordre dans lequel ils apparaissent dans la formule*. Les hypothèses sous-jacentes sont rarement celles que l'on veut tester en biologie, aussi le type I est à oublier (excepté bien sûr si c'est exactement ce que l'on souhaite tester). Les tests de type I sont réalisés par la fonction `anova(modele)`. Attention, selon les modèles ce n'est pas le même test qui est appliqué

Type II : l'analyse est *non séquentielle, i.e. indépendante de l'ordre des termes dans la formule*. Elle respecte par contre le *principe de marginalité*, qui stipule que « lorsqu'une interaction a un effet significatif, l'effet des variables de l'interaction, lorsqu'elles sont impliquées dans des termes d'ordre inférieur, est *marginal* devant celui de l'interaction ». Dit autrement, ce principe dit que si une interaction entre deux facteurs a un effet significatif, l'information est dans cette interaction et il est inutile de regarder l'effet des facteurs pris seuls (étendu à trois facteurs : si l'interaction entre les trois a un effet significatif, il est inutile de s'intéresser aux facteurs pris seuls et aux interactions d'ordre 2 impliquant ces facteurs). Concrètement, on considère donc que lorsque l'on teste l'effet d'un facteur, tous les termes d'ordre supérieur impliquant ce facteur n'ont pas d'effet. Dans notre exemple, cela donne (i) un test de l'effet de `A` après avoir retiré l'effet de `B` ; (ii) un test de l'effet de `B` après avoir retiré l'effet de `A` ; (iii) un test de l'effet de `A:B` après avoir retiré les effets de `A` et `B`. Cette approche est celle qui est systématiquement utilisée dans cet aide-mémoire, car dans la très grande majorité des cas c'est celle qui est la plus pertinente vis-à-vis des hypothèses biologiques. Les tests de type II sont réalisés par la fonction `Anova(modele,type="II")`[1] ou plus simplement `Anova(modele)`[1], le type II étant celui utilisé par défaut. Attention, selon les modèles ce n'est pas le même test qui est appliqué

Type III : l'analyse est *non séquentielle* et *ne respecte pas* le principe de marginalité. Le test de l'effet d'un facteur est donc réalisé en prenant en compte les interactions impliquant ce facteur. Dans notre exemple, cela donne (i) un test de l'effet de `A` après avoir retiré les effets de `B` et `A:B` ; (ii) un test de l'effet de `B` après avoir retiré les effets de `A` et `A:B` ; (iii) un test de l'effet de `A:B` après avoir retiré les effets de `A` et `B`. Cette approche, même si elle peut paraître intéressante au premier abord, est très délicate à utiliser car elle revient à considérer qu'un facteur puisse ne pas avoir d'effet seul tout en modifiant l'effet d'un autre facteur (*i.e.* tout en ayant un effet ailleurs). Excepté si c'est réellement l'hypothèse que l'on veut tester, le type III est donc à oublier car il est rarement pertinent vis-à-vis des hypothèses biologiques. Les tests de type III sont réalisés par la fonction `Anova(modele,type="III")`[1]. Attention, selon les modèles ce n'est pas le même test qui est appliqué.

51. Les comparaisons multiples

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lsmeans`, [2]`RVAideMemoire`

Lorsqu'un test sur un modèle a révélé un effet significatif d'un facteur (ou d'une interaction impliquant au moins un facteur), il est nécessaire de réaliser des comparaisons multiples pour voir précisément quelles sont les modalités qui diffèrent. Il existe plusieurs méthodes pour réaliser ces comparaisons, celle qui est présentée ici est l'une des plus intéressantes car elle prend en compte l'effet des autres variables explicatives du modèle dans les comparaisons. En d'autres termes les comparaisons se font après avoir retiré la variation due aux autres variables explicatives. Ce ne sont donc plus les moyennes brutes (telles qu'on peut les calculer dans la fiche **14**) qui sont comparées, mais des moyennes *ajustées* en fonction des autres variables explicatives. La méthode en question est celle des *moyennes des moindres carrés*, ou *Least Squares Means* (abrégé le plus souvent en LSMeans).

Les matrices de contrastes

Les fonctions présentées dans cette fiche permettent de réaliser de manière simple toutes les comparaisons deux-à-deux possibles. Dans les cas où l'on veut réaliser des comparaisons personnalisées, il est nécessaire de commencer par créer une *matrice des contrastes*. Celle-ci est de la forme :

	Modalité 1	Modalité 2	Modalité 3
Comparaison 1	1	-1	0
Comparaison 2	0	1	-1
Comparaison 3	2	-1	-1

Dans cette matrice, les comparaisons (ou contrastes) sont représentées en lignes, tandis que les modalités du facteur (ou de l'interaction) sont en colonnes. Les conventions d'écriture des contrastes sont les suivantes :

- les modalités n'intervenant pas dans la comparaison doivent avoir une valeur nulle
- les modalités à comparer doivent avoir une valeur non nulle et un signe opposé
- il est possible d'effectuer des regroupements de classes (comme dans la troisième ligne, où la modalité 1 est comparée au groupe formé par les modalités 2 et 3)
- la somme des valeurs positives et négatives d'un contraste doit être nulle.

Attention, **R** considère les modalités du facteur dans l'ordre alphabétique, *i.e.* la 1ère colonne correspond à la 1ère modalité dans l'ordre alphabétique, et ainsi de suite.

La matrice des contrastes est concrètement un tableau, qui peut être créé directement dans **R** (voir fiche **2**) ou importé depuis un tableur (voir fiche **10**).

Réalisation des comparaisons

Facteur seul ou interaction entre deux facteurs

Cas 1 : toutes les comparaisons deux-à-deux

La syntaxe est dans ce cas simple : `lsmeans(modele,pairwise~facteur)`[1], où `facteur` est le nom du facteur ou de l'interaction d'intérêt. Pour réaliser les comparaisons séparément pour chaque niveau d'un autre facteur : `lsmeans(modele,pairwise~facteur1|facteur2)`[1] où `facteur1` est le facteur (ou l'interaction) dont on veut comparer les modalités, et `facteur2` le facteur pour lequel on veut réaliser les comparaisons à l'intérieur de chaque modalité.

Remarque 1 : pour les GLMMs basés sur une loi binomiale négative (créés avec la fonction `glmer.nb()` ; voir fiche **57**), il est nécessaire de préciser en plus *via* l'argument `data` le tableau de données sur lequel est basé le modèle. Dans tous les autres cas ce n'est pas nécessaire.

Remarque 2 : pour pouvoir utiliser la fonction `lsmeans()`[1] sur un modèle créé avec les fonctions `clm()` et `clmm()` (voir fiche **72**) il est nécessaire d'avoir chargé le package `RVAideMemoire`.

EXEMPLE(S)

On utilise un modèle appelé `modele` ayant pour formule `reponse~A*B+C` où `A`, `B` et `C` sont des facteurs (voir fiche **48** pour une explication de la formule).

Pour comparer toutes les modalités de `A` deux-à-deux :
`> lsmeans(modele,pairwise~A)`[1]
Pour comparer les modalités de l'interaction `A:B` :
`> lsmeans(modele,pairwise~A:B)`[1]
Pour comparer les modalités de `A` séparément pour chaque modalité de `C` :
`> lsmeans(modele,pairwise~A|C)`[1]
Pour comparer les modalités de l'interaction `A:B` séparément pour chaque modalité de `C` :
`> lsmeans(modele,pairwise~A:B|C)`[1]

Si le modèle est un GLMM basé sur une loi binomiale négative et que le tableau de données s'appelle `tableau` :
`> lsmeans(modele,pairwise~A,data=tableau)`[1]

Cette procédure a l'avantage de donner la *p-value* associée à chaque comparaison, mais oblige à regrouper les modalités (dans des groupes du type a, ab, b...) à la main. Une méthode de regroupement automatique existe cependant, qui nécessite deux étapes. La première consiste à enregistrer le résultat de la fonction `lsmeans()`[1] dans un objet (appelé `LSM` ici) :
`> LSM <- lsmeans(modele,~facteur)`[1] (et éventuellement l'argument `data`)
Lorsque rien n'est précisé avant le symbole `~`, la fonction renvoie les moyennes ajustées mais ne réalise pas de comparaison.
La seconde étape consiste à réaliser toutes les comparaisons deux-à-deux et à attribuer un groupe à chaque modalité :
`> cld(LSM)`[1]

Remarque : pour pouvoir utiliser la fonction `cld()`[1] il est nécessaire d'avoir installé le package `multcompView`.

Cette méthode ne renvoyant pas la *p-value* de chaque comparaison, elle doit être vue comme complémentaire de la première.

Cas 2 : uniquement certaines comparaisons

Dans le cas de comparaisons personnalisés, la procédure se fait en deux étapes :
`> cont.lsmc <- user.cont(contrastes)`[2]
`> lsmeans(modele,cont~facteur)`[1] (et éventuellement l'argument `data`)
où `contrastes` est la matrice des contrastes et `facteur` le facteur (où l'interaction) dont on veut comparer les modalités.

Récupération des moyennes ajustées

Représenter sur un diagramme en barres les moyennes ajustées par niveau d'un facteur (ou combinaison de deux facteurs) est souvent plus pertinent que de représenter les moyennes brutes. Cela permet d'illustrer la variation réellement due à ce facteur, après avoir retiré la variation due aux autres variables explicatives du modèle.

Pour tous les modèles sauf ceux analysant des notes (voir fiche **72** pour la démarche à utiliser dans ce cas), récupérer les moyennes ajustées se fait de la même façon. Il faut préalablement avoir enregistré le résultat de la fonction `lsmeans()`[1] dans un objet (appelé `LSM` ici). Deux procédures sont ensuite possibles selon la situation :

- si aucune comparaison n'a été réalisée par la fonction `lsmeans()`[1] (*i.e.* s'il n'y a rien avant le symbole `~`) : `valeurs<-as.data.frame(summary(LSM,type="response"))`
- si des comparaisons ont été réalisées par la fonction `lsmeans()`[1] (*i.e.* s'il y a un terme avant le symbole `~`) : `valeurs<-as.data.frame(summary(LSM$lsmeans,type="response"))`

L'objet `valeurs` est un tableau dont la (les) première(s) colonne(s) correspond(ent) aux modalités du (des) facteur(s) précisé(s) dans `lsmeans()`[1]. La colonne appelée `lsmean`, `response`, `prob`, `rate` ou `hazard` contient les moyennes ajustées. La colonne `SE` contient l'erreur standard associée à chaque moyenne.

Une fois les moyennes et erreurs standards récupérées, elles peuvent être représentées sur un diagramme en barres (voir fiche **41**).

Interaction entre un facteur et une covariable

La syntaxe est du type : `fc.multcomp(modele,"interaction",contrastes)`[2] où `interaction` est le nom de l'interaction, entre guillemets. Si `contrastes` n'est pas précisé, toutes les comparaisons deux-à-deux possibles sont réalisées.

EXEMPLE(S)

On utilise un modèle appelé `modele` ayant pour formule `reponse~facteur*covariable` (voir fiche **48** pour une explication de la formule).

Pour comparer toutes les modalités de l'interaction deux-à-deux :

```
> fc.multcomp(modele,"facteur:covariable")[2]
```

Si une matrice des contrastes personnalisée a été créé :

```
> fc.multcomp(modele,"facteur:covariable",contrastes)[2]
```

52. La sélection de modèle

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]MuMIn

On utilise le plus souvent la démarche de sélection de modèle lorsque l'on a beaucoup de variables explicatives et que l'on veut identifier celles ayant le plus d'effet sur la variable à expliquer. L'objectif est d'aboutir au meilleur compromis entre un bon ajustement aux données (qui croît avec le nombre de variables explicatives) et un petit nombre de variables explicatives (plus on prend en compte de variables explicatives, plus on perd en capacité à généraliser les résultats).

La sélection est basée sur une valeur (appelée *critère*) représentant ce compromis. Plus elle est faible, meilleur est le compromis. Il existe plusieurs critères, le plus utilisé est le Critère d'Information d'Akaike (AIC).

En pratique, la sélection est réalisée automatiquement à partir du modèle initial (complet). Pour la réaliser : `dredge(modele)`[1], où `modele` est le modèle contenant toutes les variables explicatives. Cette fonction calcule l'AIC de tous les modèles possibles à partir du modèle initial et renvoie un tableau classant tous ces modèles. Parmi les arguments facultatifs de la fonction, deux sont particulièrement intéressants :

- `m.max=valeur`, où `valeur` est le nombre maximal de variables explicatives à intégrer dans les modèles à tester
- `fixed=variables`, où `variables` est un vecteur contenant le nom des variables explicatives à intégrer dans tous les modèles à tester (entre guillemets).

Attention, si $\frac{n}{df} < 40$, où n est le nombre d'individus et df le nombre de paramètres estimés par modèle (renvoyé par le fonction `dredge()`[1]), il faut utiliser l'AICc (AIC corrigé) et non pas l'AIC. En fait, comme cette situation est courante, la fonction `dredge()` utilise par défaut l'AICc pour classer les modèles.

Dans le cas de modèles avec une loi `quasipoisson` ou `quasibinomial`, le critère utilisé est le QAIC, dérivé de l'AIC pour les distributions « quasi ». Il est nécessaire pour ces modèles de renseigner la valeur du paramètre de dispersion (obtenu *via* `summary(modele)`) grâce à l'argument `chat`.

Pour utiliser un autre critère que l'AICc pour réaliser la sélection, utiliser l'argument `rank` (en donnant comme valeur `"AIC"` ou `"QAIC"` selon les cas).

— EXEMPLE(S) —

Avec un modèle du type :

```
> modele <- glm(formule,family="quasipoisson")
```

La valeur du paramètre de dispersion est récupérée *via* :

```
> summary(modele)
[...]
(Dispersion parameter for quasipoisson family taken to be 1.126135)
[...]
```

Et la sélection automatique peut ensuite être réalisée :

```
> dredge(modele,rank="QAIC",chat=1.126135)[1]
```

La démarche est identique pour les modèles à loi `quasibinomial`.

*Note sur les Modèles Linéaires Mixtes (*LMM*)* : le modèle complet doit avoir été créé avec l'option `REML=FALSE` (par défaut `REML=TRUE`). Une fois le meilleur modèle déterminé, il doit être récrit avec l'option `REML=TRUE` (ou en ne précisant pas l'option `REML`) pour être analysé.

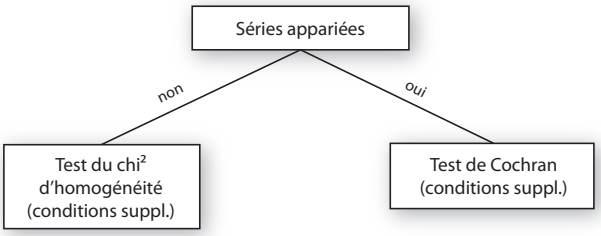
Lorsque l'on utilise une procédure de sélection automatique, il faut être conscient que le modèle avec l'AIC (ou dérivé) le plus faible n'est pas forcément celui qui *biologiquement* a le plus de sens. Il ne faut donc pas utiliser cette procédure aveuglément mais toujours réfléchir aux variables et interactions qui sont retenues. Il est ainsi possible d'enlever des termes manuellement si ceux-ci ne sont pas pertinents ou trop complexes à interpréter (comme une interaction d'ordre 3 ou 4).

53. Comparaison de plusieurs probabilités de réponse

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Séries non appariées

Test du χ^2 d'homogénéité (ou d'indépendance ; non paramétrique)

Conditions : les effectifs théoriques doivent tous être non nuls et 80 % d'entre eux doivent être ≥ 5 (voir ci-dessous).

Pour réaliser le test : `chisq.bintest(variable~facteur)`[1] où `variable` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

La fonction teste automatiquement la double condition sur les effectifs théoriques, appelée « règle de Cochran ». Si ces conditions sont respectées un test du χ^2 d'homogénéité est utilisé, dans le cas contraire un test exact de Fisher est utilisé.

Si la *p-value* du test est significative, cela indique qu'au moins deux classes du facteur ont un effet différent sur la variable à expliquer (sans préciser lesquelles). La fonction réalise alors automatiquement toutes les comparaisons deux-à-deux possibles, soit par un test du χ^2 d'homogénéité soit par un test exact de Fisher.

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles probabilités sont responsables du rejet de l'hypothèse nulle dans le test global.

Séries appariées

Test Q de Cochran (non paramétrique)

Conditions : le plan d'expérience doit être en blocs aléatoires complets sans répétition (voir fiche **7**).

Pour réaliser le test : `cochran.qtest(reponse~fact.fixe|fact.alea)`[1] où `variable`, `fact.fixe` et `fact.alea` sont des vecteurs contenant la valeur de chaque individu (dans le même ordre) pour la variable à expliquer, le facteur (fixe) dont on veut comparer les modalités et le facteur (aléatoire) servant à définir les séries appariées, respectivement.

Si la *p-value* du test est significative, cela indique qu'au moins deux classes du facteur fixe ont un effet différent sur la variable à expliquer (sans préciser lesquelles). La fonction réalise alors automatiquement toutes les comparaisons deux-à-deux possibles par le test des signes de Wilcoxon.

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles probabilités sont responsables du rejet de l'hypothèse nulle dans le test global.

54. Analyser des probabilités de réponse

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lme4`, [2]`car`, [3]`RVAideMemoire`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **47** à **50**. ⚠

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM), tandis qu'avec des séries appariées c'est un Modèle Linéaire Généralisé Mixte (*Generalized Linear Mixed Model* ou GLMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **5**)).

Contrairement au Modèle Linéaire (et sa variante Mixte) utilisé pour analyser des moyennes, les GLM(M)s ne sont pas basés sur une loi normale. Dans le cas de probabilités de réponse, la loi à utiliser est une loi binomiale.

Remarque : il existe en fait plusieurs types de GLM(M)s basés sur une loi binomiale. Le cas décrit ici – le plus fréquent – est celui d'un modèle *logistique*. Lorsque toutes les variables explicatives sont des covariables, on est dans le cas particulier d'une *régression logistique*.

Construction du modèle

La variable à expliquer peut être codée numériquement (sous forme 0/1) ou être un facteur à deux niveaux.

Pour créer le modèle :

- sans séries appariées : `modele<-glm(formule,family="binomial")`
- avec des séries appariées : `modele<-glmer(formule,family="binomial")`[1].

Voir fiche **48** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

- inclure un facteur permet de tester si les probabilités de réponse diffèrent entre les niveaux de ce facteur
- inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer
- inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Voir fiche **49** pour une explication détaillée de cette vérification.

Test(s)

Quel que soit le modèle, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[2] (voir fiche **50** pour une explication détaillée des hypothèses testées). Cependant, ce ne sont pas les mêmes tests qui sont réalisés selon le modèle :

- GLM : la fonction réalise un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé)
- GLMM : la fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui

différent. Voir fiche **51** pour réaliser ces comparaisons.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une probabilité de réponse nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction (seule la seconde est disponible pour les GLMMs), les deux étant basées sur la fonction `predict()` :

- donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables),type="response")`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`
- créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau,type="response")`.
  Remarque : pour les GLMMs, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument `REform=NA` à la fonction `predict()`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- glm(reponse~facteur*covariable,family="binomial")
```

On peut prédire une probabilité de réponse de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10),type="response")
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)),type="response")
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)),type="response")
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau,type="response")
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres. Deux types de moyennes peuvent être représentées :

- les moyennes brutes (*i.e.* calculées à partir des données brutes), avec leurs erreurs standards : voir fiches **14** et **40**, et l'utilisation de la fonction `tapply()`. Attention, dans le cas des GLMMs, ces moyennes et erreurs standards ne tiennent pas compte du (des) facteur(s) aléatoire(s)
- les moyennes ajustées en fonction des autres variables du modèle, également avec leurs erreurs standards : voir fiche **51**.

Une fois les moyennes et erreurs standards récupérées, le diagramme peut être tracé (voir fiche **41**).

Relation avec une covariable

Le modèle logistique suppose une relation sigmoïde (*i.e.* en « S ») entre la variable à expliquer et chacune des covariables. Illustrer cette relation nécessite trois étapes :

1. tracer les points observés : `plot(reponse~covariable)`
2. créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[3]
3. ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer,type="response"))`.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- glm(reponse~facteur*covariable,family="binomial")
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(reponse~covariable)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(covariable)[3]
```

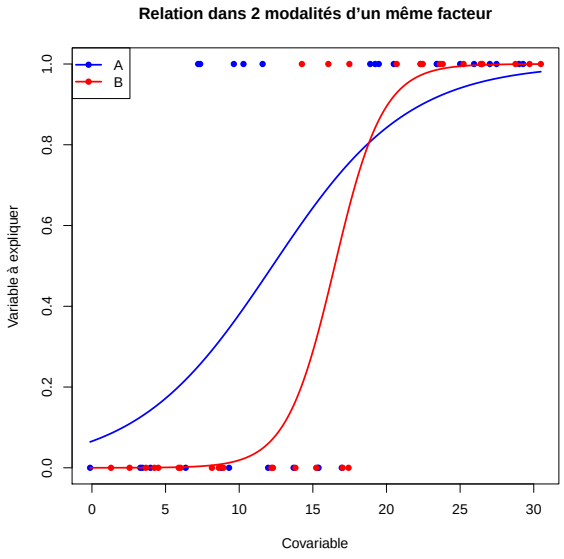
Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x))),
type="response"))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x))),
type="response"))
```



55. Conformité de plusieurs effectifs avec des valeurs théoriques

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Test du χ^2 de conformité (non paramétrique)

Conditions : les effectifs théoriques doivent tous être non nuls et 80 % d'entre eux doivent être ≥ 5 (« règle de Cochran », voir ci-dessous).

Les effectifs théoriques sont donnés par la fonction `chisq.test(effectifs,p=prop.theo)$expected`, où `effectifs` est un vecteur contenant les effectifs et `prop.theo` un vecteur contenant les proportions théoriques (et *non* les effectifs théoriques), dans le même ordre qu'`effectifs`. La somme de ces proportions doit être égale à 1. Ces effectifs théoriques permettent de vérifier si la règle de Cochran est respectée.

— EXEMPLE(S) —

```
> effectifs <- c(30,48,22)
> effectifs.theo <- c(25,50,25)
> prop.theo <- effectifs.theo / sum(effectifs.theo)
> prop.theo
[1] 0.25 0.50 0.25
> sum(prop.theo)
[1] 1
> chisq.test(effectifs,p=prop.theo)$expected
[1] 25 50 25
```

Pour réaliser le test : `chisq.test(effectifs,p=prop.theo)`.

Remarque : la fonction `chisq.test()` n'utilise pas la règle de Cochran et renvoie un avertissement dès qu'au moins un effectif théorique est < 5. Si la règle est bien respectée, ne pas tenir compte de l'avertissement.

Une *p-value* significative indique qu'au moins un effectif diffère de sa valeur théorique, sans préciser le(s)quel(s). Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier l'(les) effectif(s) en question, *via* `chisq.theo.multcomp(effectifs,p=prop.theo)`[1].

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quel effectif est responsable du rejet de l'hypothèse nulle dans le test global.

56. Comparaison de plusieurs effectifs – sans répétition

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

On dit qu'il n'y a pas de répétition lorsque l'on a un effectif par modalité .

Test du χ^2 d'homogénéité (non paramétrique)

Conditions : les effectifs théoriques doivent tous être non nuls et 80 % d'entre eux doivent être ≥ 5 (« règle de Cochran », voir ci-dessous).

Les effectifs théoriques sont donnés par la fonction `chisq.test(effectifs)$expected`, où `effectifs` est un vecteur contenant les effectifs. Ces effectifs théoriques permettent de vérifier si la règle de Cochran est respectée.

Pour réaliser le test : `chisq.test(effectifs)`.

Remarque : la fonction `chisq.test()` n'utilise pas la règle de Cochran et renvoie un avertissement dès qu'au moins un effectif théorique est < 5. Si la règle est bien respectée, ne pas tenir compte de l'avertissement.

Une *p-value* significative indique qu'au moins deux effectifs diffèrent l'un de l'autre, sans préciser lesquels. Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les effectifs en question, *via* `chisq.multcomp(effectifs)`[1].

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quels effectifs sont responsables du rejet de l'hypothèse nulle dans le test global.

57. Analyser des effectifs

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lme4`, [2]`RVAideMemoire`, [3]`MASS`, [4]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **47** à **50**. ⚠

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM), tandis qu'avec des séries appariées c'est un Modèle Linéaire Généralisé Mixte (*Generalized Linear Mixed Model* ou GLMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **5**)).

Contrairement au Modèle Linéaire (et sa variante Mixte) utilisé pour analyser des moyennes, les GLM(M)s ne sont pas basés sur une loi normale. Dans le cas d'effectifs, la loi à utiliser *a priori* est une loi de Poisson.

Remarque : il existe en fait plusieurs types de GLM(M)s basés sur une loi de Poisson. Le cas décrit ici – le plus fréquent – est celui d'un modèle *log-linéaire*. Lorsque toutes les variables explicatives sont des covariables, on est dans le cas particulier d'une *régression log-linéaire* ou *régression de Poisson*.

Construction du modèle

Pour créer le modèle :

- sans séries appariées : `modele<-glm(formule,family="poisson")`
- avec des séries appariées : `modele<-glmer(formule,family="poisson")`[1].

Voir fiche **48** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

- inclure un facteur permet de tester si les effectifs diffèrent entre les niveaux de ce facteur
- inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer
- inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de la validité du modèle

Surdispersion des résidus

La première étape de validation du modèle est de vérifier que les résidus ne sont pas *surdispersés, i.e.* que leur variance n'est pas plus grande que celle admise par le modèle. Pour cela, il suffit de comparer la *déviance résiduelle* (*residual deviance*) du modèle avec ses *degrés de liberté résiduels* (*residual degrees of freedom*). Si la déviance résiduelle est plus grande, il y a surdispersion. Pour obtenir ces valeurs :

- GLM : appeler `summary(modele)` et repérer la ligne `Residual deviance: xx.xx on xx degrees of freedom`
- GLMM : utiliser `overdisp.glmer(modele)`[2].

S'il y a surdispersion, il est nécessaire de modifier le modèle en changeant la loi sur laquelle il est basé :

- GLM : deux solutions sont envisageables :
  - remplacer la loi de Poisson par une loi quasi-Poisson. Il faut recréer le modèle : `modele<-glm(formule,family="quasipoisson")`
  - remplacer la loi de Poisson par une loi binomiale négative. Il faut recréer le modèle, cette fois avec une autre fonction : `modele<-glm.nb(formule)`[3]

  Pour choisir entre les deux solutions, la meilleure chose à faire est de créer les deux modèles et de garder celui qui est le mieux ajusté aux données (voir fiche **49**)

– GLMM : seule la loi binomiale négative est disponible. Il faut recréer le modèle, avec une autre fonction : `modele<-glmer.nb(formule)`[1].

Ajustement aux données

La seconde vérification, indispensable, est de contrôler que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Voir fiche **49** pour une explication détaillée de cette vérification.

Test(s)

Pour tous les modèles évoqués à l'exception de ceux avec une loi quasi-Poisson, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[4] (voir fiche **50** pour une explication détaillée des hypothèses testées). Cependant, ce ne sont pas les mêmes tests qui sont réalisés selon le modèle :

– GLM : la fonction réalise un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé)
– GLMM : la fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Dans le cas d'un modèle avec une loi quasi-Poisson, le test à utiliser est un test F. Il est réalisé *via* `Anova(modele,test="F")`[4].

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui différent. Voir fiche **51** pour réaliser ces comparaisons.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire un effectif nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction (seule la seconde est disponible pour les GLMMs), les deux étant basées sur la fonction `predict()` :

– donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables),type="response")`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`
– créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau,type="response")`.
Remarque : pour les GLMMs, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument `REform=NA` à la fonction `predict()`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- glm(reponse~facteur*covariable,family="poisson")
```

On peut prédire un effectif de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10),type="response")
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)),type="response")
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)),type="response")
```

Ou encore créer un tableau de ce type :

```
> tableau
     facteur   covariable
 1         A           10
 2         B           10
```

Puis :

```
> predict(modele,newdata=tableau,type="response")
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres. Deux types de moyennes peuvent être représentées :

- les moyennes brutes (*i.e.* calculées à partir des données brutes), avec leurs erreurs standards : voir fiches **14** et **40**, et l'utilisation de la fonction `tapply()`. Attention, dans le cas des GLMMs, ces moyennes et erreurs standards ne tiennent pas compte du (des) facteur(s) aléatoire(s)
- les moyennes ajustées en fonction des autres variables du modèle, également avec leurs erreurs standards : voir fiche **51**.

Une fois les moyennes et erreurs standards récupérées, le diagramme peut être tracé (voir fiche **41**).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. tracer les points observés : `plot(reponse~covariable)`
2. créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[2]
3. ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer,type="response"))`.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- glm(reponse~facteur*covariable,family="poisson")
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(reponse~covariable)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(covariable)
```

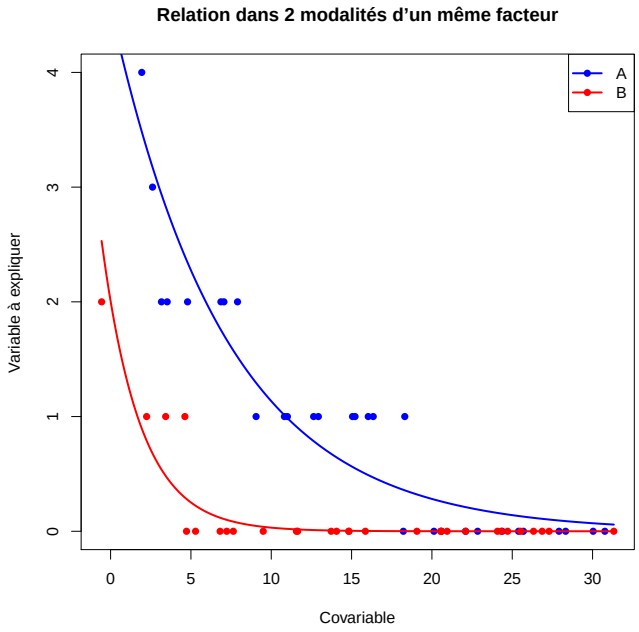
Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x))),
type="response"))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x))),
type="response"))
```



58. Conformité d'une proportion avec une valeur théorique

Test binomial exact (non paramétrique)

Pour réaliser le test : `binom.test(n1,n,p=prop.theo)` où `n1` est le nombre d'individus de la catégorie d'intérêt, `n` l'effectif total et `prop.theo` la proportion théorique de la catégorie d'intérêt.

EXEMPLE(S)

On veut comparer le sex-ratio (*i.e.* la proportion de femelles) d'un échantillon de 20 individus contenant 7 femelles et 13 mâles à un sex-ratio équilibré (donc une proportion de 0,5) :

```
> binom.test(7,20,p=0.5)
```

59. Conformité de plusieurs proportions avec des valeurs théoriques

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Les données doivent être présentées sous la forme de deux facteurs : l'un définissant les k modalités à tester, l'autre définissant les deux groupes à l'intérieur de chaque modalité (ex : réussite/échec, vivant/mort, femelle/mâle...). Grâce à ces deux variables, les données peuvent (et doivent) être organisées en un tableau de contingence du type :

		Variable B (groupes)	
		Classe 1	Classe 2
Variable A (modalités à tester)	Classe 1		
	...		
	Classe k		

où chaque case contient le nombre d'individus possédant à la fois le caractère de la variable A et celui de la variable B.

Ce tableau est obtenu de la manière suivante : `tab.cont<-table(variableA,variableB)` où `variableA` et `variableB` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre).

Les proportions comparées sont celles de la première colonne du tableau de contingence (*i.e.* la Classe 1 de la variable B).

Test du χ^2 (non paramétrique)

Conditions : les effectifs théoriques doivent tous être non nuls et 80 % d'entre eux doivent être ≥ 5 (« règle de Cochran », voir ci-dessous).

Les effectifs théoriques sont obtenus *via* `chisq.exp(tab.cont,p=prop.theo)`[1] où `prop.theo` est un vecteur contenant la proportion théorique dans chaque modalité (de 1 à k). Ils permettent de vérifier si la règle de Cochran est respectée.

Pour réaliser le test : `prop.test(tab.cont,p=prop.theo)`.

Remarque : la fonction `prop.test()` n'utilise pas la règle de Cochran et renvoie un avertissement dès qu'au moins un effectif théorique est < 5. Si la règle est bien respectée, ne pas tenir compte de l'avertissement.

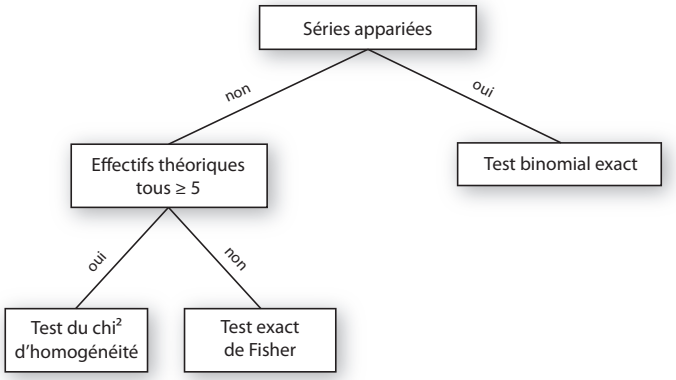
Une *p-value* significative indique qu'au moins une proportion diffère de sa valeur théorique, sans préciser la(les)quelle(s). Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier la (les) proportion(s) en question, *via* `prop.multcomp(tab.cont,p=prop.theo)`[1].

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelle modalité est responsable du rejet de l'hypothèse nulle dans le test global.

60. Comparaison de deux proportions – sans répétition

On dit qu'il n'y a pas de répétition lorsque l'on a une valeur de la proportion en question par modalité (*i.e.* un échantillon par population).

Pour choisir le test approprié :



Séries non appariées

Les données doivent être présentées sous la forme de deux facteurs : l'un définissant les deux modalités à comparer, l'autre définissant les deux groupes à l'intérieur de chaque modalité (ex : réussite/échec, vivant/mort, femelle/mâle...). Grâce à ces deux variables, les données peuvent (et doivent) être organisées en un tableau de contingence du type :

		Variable B (groupes)	
		Classe 1	Classe 2
Variable A	Classe 1		
(à comparer)	Classe 2		

où chaque case contient le nombre d'individus possédant à la fois le caractère de la variable A et celui de la variable B.

Ce tableau est obtenu de la manière suivante : `tab.cont<-table(variableA,variableB)` où `variableA` et `variableB` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre).

Dans les tests suivants, les proportions comparées sont celles de la première colonne du tableau de contingence (*i.e.* la Classe 1 de la variable B).

Les effectifs théoriques sont donnés par la fonction `chisq.test(tab.cont)$expected`. La fonction renvoie un avertissement si la condition requise par le test du χ^2 d'homogénéité (*i.e.* effectifs théoriques tous ≥ 5) n'est pas remplie.

Test du χ^2 d'homogénéité (ou d'indépendance ; non paramétrique)

Pour réaliser le test : `prop.test(tab.cont)`.

Test exact de Fisher (non paramétrique)

Pour réaliser le test : `fisher.test(tab.cont)`.

Séries appariées

Dans le cas de séries appariées, les individus (au sens statistique) sont reliés par paire. Ce peuvent être des entités réellement différentes, ou un même individu mesuré deux fois.

Les données doivent donc être présentées différemment, sous la forme de deux facteurs : l'un définissant la mesure du 1er individu de chaque paire (ex : réussite/échec, vivant/mort, femelle/mâle...), l'autre définissant la mesure du 2nd individu de chaque paire. Le tableau de contingence est obtenu *via* `tab.cont<-table(individus1,individus2)` où `individus1` est un vecteur contenant la valeur du 1er individu de chaque paire, et `individus2` un vecteur contenant la valeur du 2nd individu de chaque paire (dans le même ordre qu'`individus1`) :

		Individus 2	
		Classe 1	Classe 2
Individus1	Classe 1		
	Classe 2		

Test binomial exact (non paramétrique)

Pour réaliser le test : `binom.test(n1.2,ns)` où `n1.2` est le nombre d'individus dans la case en haut à droite (*i.e.* paires dont le 1er individu est de classe 1 et le 2nd de classe 2), et `ns` est le nombre de paires d'individus dont les valeurs diffèrent (*i.e.* case en bas à gauche + case en haut à droite).

61. Comparaison de plus de deux proportions – sans répétition

On dit qu'il n'y a pas de répétition lorsque l'on a une valeur de la proportion en question par modalité (*i.e.* un échantillon par population).

Les données doivent être présentées sous la forme de deux facteurs : l'un définissant les k modalités à comparer, l'autre définissant les deux groupes à l'intérieur de chaque modalité (ex : réussite/échec, vivant/mort, femelle/mâle...). Grâce à ces deux variables, les données peuvent (et doivent) être organisées en un tableau de contingence du type :

		Variable B (groupes)	
		Classe 1	Classe 2
Variable A (modalités à comparer)	Classe 1		
	...		
	Classe k		

où chaque case contient le nombre d'individus possédant à la fois le caractère de la variable A et celui de la variable B.

Ce tableau est obtenu de la manière suivante : `tab.cont<-table(variableA,variableB)` où `variableA` et `variableB` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre).

Les proportions comparées sont celles de la première colonne du tableau de contingence (*i.e.* la Classe 1 de la variable B).

Test du χ^2 d'homogénéité (ou d'indépendance ; non paramétrique)

Conditions : les effectifs théoriques doivent tous être non nuls et 80 % d'entre eux doivent être ≥ 5 (« règle de Cochran », voir ci-dessous).

Les effectifs théoriques sont donnés par la fonction `chisq.test(tab.cont)$expected`. Ils permettent de vérifier si la règle de Cochran est respectée.

Pour réaliser le test : `prop.test(tab.cont)`.

Remarque : la fonction `prop.test()` n'utilise pas la règle de Cochran et renvoie un avertissement dès qu'au moins un effectif théorique est < 5. Si la règle est bien respectée, ne pas tenir compte de l'avertissement.

Une *p-value* significative indique qu'au moins deux proportions diffèrent l'une de l'autre, sans préciser lesquelles. Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les proportions en question, *via* `pairwise.prop.test(tab.cont)`.

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles proportions sont responsables du rejet de l'hypothèse nulle dans le test global.

62. Analyser des proportions

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lme4`, [2]`RVAideMemoire`, [3]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **47** à **50**. ⚠

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM), tandis qu'avec des séries appariées c'est un Modèle Linéaire Généralisé Mixte (*Generalized Linear Mixed Model* ou GLMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **5**)).

Contrairement au Modèle Linéaire (et sa variante Mixte) utilisé pour analyser des moyennes, les GLM(M)s ne sont pas basés sur une loi normale. Dans le cas de proportions, la loi à utiliser *a priori* est une loi binomiale.

Remarque : il existe en fait plusieurs types de GLM(M)s basés sur une loi binomiale. Le cas décrit ici – le plus fréquent – est celui d'un modèle *logistique*. Lorsque toutes les variables explicatives sont des covariables, on est dans le cas particulier d'une *régression logistique*.

Construction du modèle

Les modèles d'analyse de proportions sont des cas particuliers où la variable à expliquer n'est pas un vecteur, mais une matrice à deux colonnes du type :

```
     Groupe 1  Groupe 2
[1,]       10        16
[2,]        8        24
[3,]       14        11
...
```

Dans cette matrice, chaque ligne correspond à un échantillon et chaque colonne à un groupe à l'intérieur des échantillons (les deux colonnes définissent donc la proportion étudiée ; ex : réussite/échec, vivant/mort, femelle/mâle...). Les données dans cette matrice sont des effectifs, *i.e.* chaque case contient le nombre d'individus appartenant à la fois à l'échantillon `[x,]` (où `x` est le numéro de la ligne) et au groupe (colonne) correspondant. Au sens statistique, un individu est représenté par une *ligne* du tableau. Les proportions sur lesquelles le modèle est basé sont celles de la première colonne de la matrice (*i.e.* le Groupe 1).

Cette matrice peut être obtenue *via* `reponse<-cbind(groupe1,groupe2)` où `groupe1` et `groupe2` sont des vecteurs correspondant aux deux colonnes (la 1ère valeur correspondant à la 1ère ligne de la matrice et les deux vecteurs étant dans le même ordre). Dans la formule du modèle, la variable à expliquer est `reponse`.

Pour ensuite créer le modèle :

- sans séries appariées : `modele<-glm(formule,family="binomial")`
- avec des séries appariées : `modele<-glmer(formule,family="binomial")`[1].

Voir fiche **48** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

- inclure un facteur permet de tester si les proportions diffèrent entre les niveaux de ce facteur
- inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer

– inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de la validité du modèle

Surdispersion des résidus

La première étape de validation du modèle est de vérifier que les résidus ne sont pas *surdispersés, i.e.* que leur variance n'est pas plus grande que celle admise par le modèle. Pour cela, il suffit de comparer la *déviance résiduelle* (*residual deviance*) du modèle avec ses *degrés de liberté résiduels* (*residual degrees of freedom*). Si la déviance résiduelle est plus grande, il y a surdispersion. Pour obtenir ces valeurs :

– GLM : appeler `summary(modele)` et repérer la ligne `Residual deviance: xx.xx on xx degrees of freedom`
– GLMM : utiliser `overdisp.glmer(modele)`[2].

S'il y a surdispersion, il est nécessaire de modifier le modèle :

– GLM : la méthode consiste à modifier la loi sur laquelle est basée le modèle, en remplaçant la loi binomiale par une loi quasi-binomiale. Il faut pour cela recréer le modèle : `modele<-glm(formule,family="quasibinomial")`
– GLMM : la méthode consiste à ajouter un facteur aléatoire au modèle, dont chaque niveau correspond à une observation. Cela se fait en deux étapes :
  1. créer le facteur aléatoire : `obs<-factor(1:nrow(reponse))`
  2. recréer le modèle en ajoutant à la fin de la formule `+(1|obs)`.

Ajustement aux données

La seconde vérification, indispensable, est de contrôler que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Voir fiche **49** pour une explication détaillée de cette vérification.

Test(s)

Pour tous les modèles évoqués à l'exception de ceux avec une loi quasi-binomiale, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[3] (voir fiche **50** pour une explication détaillée des hypothèses testées). Cependant, ce ne sont pas les mêmes tests qui sont réalisés selon le modèle :

– GLM : la fonction réalise un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé)
– GLMM : la fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Dans le cas d'un modèle avec une loi quasi-binomiale, le test à utiliser est un test F. Il est réalisé *via* `Anova(modele,test="F")`[3].

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **51** pour réaliser ces comparaisons.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une proportion nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction (seule la seconde est disponible pour les GLMMs), les deux étant basées sur la fonction `predict()` :

– donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables),type="response")`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`

– créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau,type="response")`.
Remarque : pour les GLMMs, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument `REform=NA` à la fonction `predict()`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- glm(reponse~facteur*covariable,family="binomial")
```

On peut prédire une proportion de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10),type="response")
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)),type="response")
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)),type="response")
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau,type="response")
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres. Deux types de moyennes peuvent être représentées :

– les moyennes brutes (*i.e.* calculées à partir des données brutes), avec leurs erreurs standards : voir fiches **14** et **40**, et l'utilisation de la fonction `tapply()`. Attention, dans le cas des GLMMs, ces moyennes et erreurs standards ne tiennent pas compte du (des) facteur(s) aléatoire(s).
Remarque : la fonction `tapply()` fonctionne à partir d'un vecteur de valeurs numériques. Ici la variable à expliquer est une matrice à deux colonnes, ce qui n'est pas adapté. Il faut donc préalablement calculer manuellement les proportions : `proportions<-groupe1/(groupe1+groupe2)`. C'est sur ce vecteur `proportions` que doit être utilisée la fonction `tapply()`
– les moyennes ajustées en fonction des autres variables du modèle, également avec leurs erreurs standards : voir fiche **51**.

Une fois les moyennes et erreurs standards récupérées, le diagramme peut être tracé (voir fiche **41**).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. tracer les points observés : `plot(proportions~covariable)`
2. créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[2]
3. ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée

de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer,type="response"))`.
Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- glm(reponse~facteur*covariable,family="binomial")
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(proportions~covariable)
```

Étape 2 : créer le vecteur `x` :

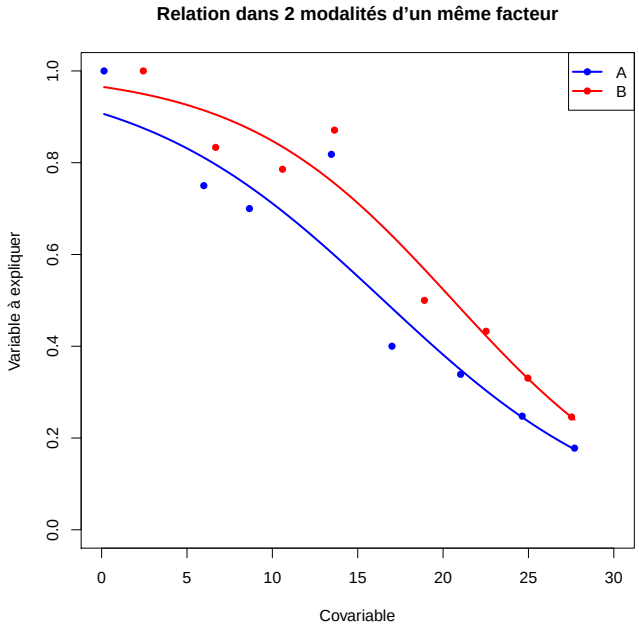
```
> x <- seq2(covariable)[2]
```

Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x))),
type="response"))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.
Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x))),
type="response"))
```

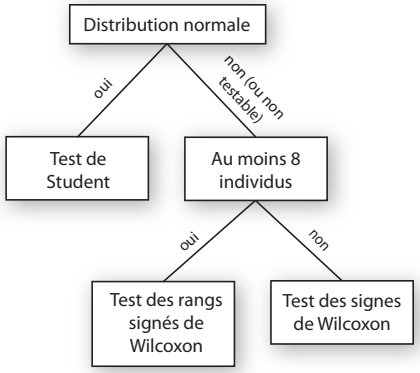


63. Conformité d'une moyenne avec une valeur théorique

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Test t *de Student (paramétrique)*

Pour réaliser le test : `t.test(serie,mu=m.theo)` où `serie` est un vecteur contenant la série de données et `m.theo` la moyenne théorique à comparer.

Test des rangs signés de Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.test(serie,mu=m.theo)`.

Ce test compare en fait la *médiane* de l'échantillon avec la valeur théorique. Mais si la distribution des données est *symétrique* (uni- ou polymodale, peu importe), médiane et moyenne sont très proches et la conclusion du test peut donc s'appliquer à la moyenne de l'échantillon. Examiner cette condition à l'aide d'un graphe de type histogramme (voir fiche **12**).

Test des signes de Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.signtest(serie,mu=m.theo)`[1].

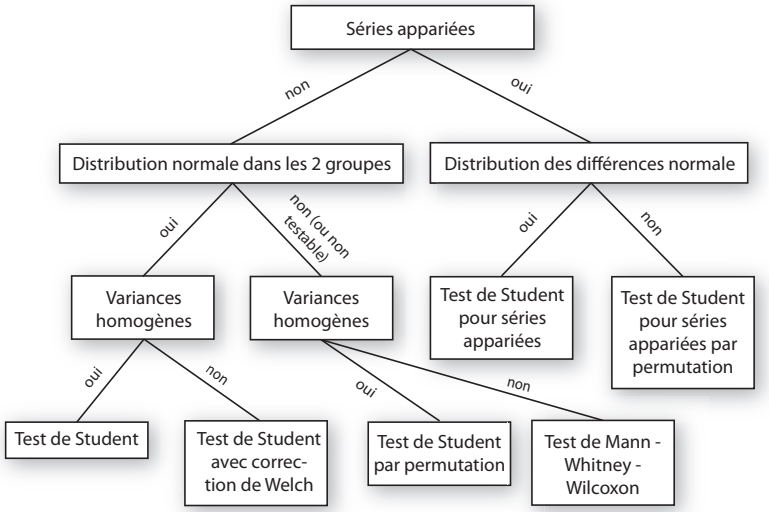
Ce test compare en fait la *médiane* de l'échantillon avec la valeur théorique. Sa conclusion ne peut donc s'appliquer à la moyenne de l'échantillon que si la distribution des données est symétrique.

64. Comparaison de deux moyennes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Séries non appariées

Test de Student (paramétrique)

Pour réaliser le test : `t.test(variable~facteur,var.equal=TRUE)`, où `variable` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Test de Student avec correction de Welch (paramétrique)

Pour réaliser le test : `t.test(variable~facteur,var.equal=FALSE)` ou plus simplement `t.test(variable~facteur)` (la correction est en fait appliquée par défaut).

Test de Student par permutation (non paramétrique)

Pour réaliser le test : `perm.t.test(variable~facteur)`[1].

Test de Mann - Whitney - Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.test(variable~facteur)`.

Ce test compare en fait la *médiane* des deux échantillons. Mais si la distribution des données a la même forme (peu importe celle-ci) dans les deux classes du facteur, la conclusion du test peut s'appliquer à leurs moyennes. Examiner cette condition à l'aide d'un graphe de type histogramme (voir fiche **12**).

Séries appariées

Pour obtenir la différence entre les valeurs appariées de chaque groupe, passer par `differences<-variable[as.numeric(facteur)==1]-variable[as.numeric(facteur)==2]`.

Test de Student pour séries appariées (paramétrique)

Pour réaliser le test : `t.test(variable~facteur,paired=TRUE)`.

Test de Student pour séries appariées par permutation (non paramétrique)

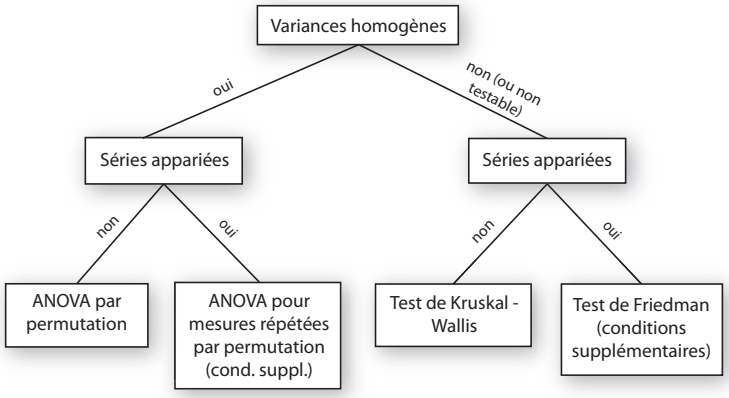
Pour réaliser le test : `perm.t.test(variable~facteur,paired=TRUE)`[1].

65. Comparaison de plus de deux moyennes – un facteur

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Remarque : la célèbre ANOVA et son équivalent pour séries appariées (ou mesures répétées) sont en fait des tests basés sur un modèle. Pour les réaliser, voir fiche **67**.

Séries non appariées

ANOVA par permutation (non paramétrique)

Pour réaliser le test : `perm.anova(variable~facteur)`[1], où `variable` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Une *p-value* significative indique qu'au moins deux classes du facteur ont un effet différent sur la variable à expliquer (sans préciser lesquelles). Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les classes en question, *via* `pairwise.perm.t.test(variable,facteur)`[1].

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles moyennes sont responsables du rejet de l'hypothèse nulle dans le test global.

Test de Kruskal - Wallis (non paramétrique)

Pour réaliser le test : `kruskal.test(variable~facteur)`.

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `pairwise.wilcox.test(variable,facteur)`.

Ces tests comparent en fait la *médiane* des différents échantillons. Mais si la distribution des données a la même forme (peu importe celle-ci) dans tous les groupes, la conclusion du test peut s'appliquer à leurs moyennes. Examiner cette condition à l'aide d'un graphe de type histogramme (voir fiche **12**).

Séries appariées

ANOVA pour mesures répétées par permutation (non paramétrique)

Conditions supplémentaires : le plan d'expérience doit être équilibré (*i.e.* contenir autant d'individus dans chaque combinaison du facteur fixe et du facteur aléatoire).

Pour réaliser le test : `perm.anova(variable~fact.fixe|fact.alea)`[1] où `variable`, `fact.fixe` et `fact.alea` sont des vecteurs contenant la valeur de chaque individu (dans le même ordre) pour la variable à expliquer, le facteur (fixe) dont on veut comparer les modalités et le facteur (aléatoire) servant à définir les séries appariées, respectivement.

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `wilcox.paired.multcomp(variable~fact.fixe|fact.alea)`[1].

Test de Friedman (non paramétrique)

Conditions supplémentaires : le plan d'expérience doit être en blocs aléatoires complets sans répétition (voir fiche **7**).

Pour réaliser le test : `friedman.test(variable~fact.fixe|fact.alea)`.

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `wilcox.paired.multcomp(variable~fact.fixe|fact.alea)`[1].

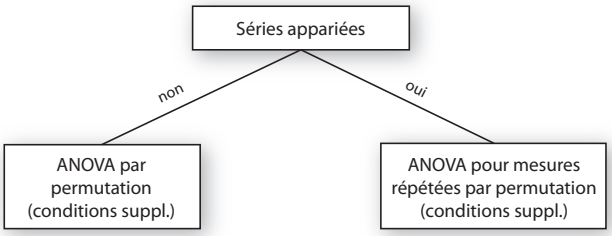
Ces tests comparent en fait la *médiane* des différents échantillons. Mais si la distribution des données a la même forme (peu importe celle-ci) dans tous les groupes, la conclusion du test peut s'appliquer à leurs moyennes. Examiner cette condition à l'aide d'un graphe de type histogramme (voir fiche **12**).

66. Comparaison de plus de deux moyennes – deux facteurs

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Pour choisir le test approprié :



Remarque : la célèbre ANOVA et son équivalent pour séries appariées (ou mesures répétées) sont en fait des tests basés sur un modèle. Pour les réaliser, voir fiche **67**.

Séries non appariées

ANOVA par permutation (non paramétrique)

Conditions supplémentaires : le plan d'expérience doit être équilibré (*i.e.* contenir autant d'individus dans chaque combinaison des deux facteurs).

Pour réaliser le test : `perm.anova(formule)`[1] où `formule` peut être (voir fiche **48** pour une explication détaillée de la construction d'une formule) :

- `reponse~fact1+fact2`
- `reponse~fact1*fact2`
- `reponse~fact1/fact2` pour un modèle où *les deux facteurs sont fixes*
- `reponse~fact1/fact2` pour un modèle où `fact2` *est un facteur aléatoire*, en ajoutant l'argument `nest.f2="random"`.

Si une *p-value* est significative, cela indique qu'au moins deux classes du facteur en question (ou au moins deux combinaisons de classes des deux facteurs si c'est l'effet de l'interaction qui est significatif) ont un effet différent sur la variable à expliquer (sans préciser lesquelles). Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les classes (ou combinaisons de classes) en question, *via* `pairwise.perm.t.test(reponse,facteur)`[1], où `facteur` est `fact1`, `fact2` ou `fact1:fact2` selon la *p-value* qui est significative.

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles moyennes sont responsables du rejet de l'hypothèse nulle dans le test global.

Séries appariées

ANOVA pour mesures répétées par permutation (non paramétrique)

Conditions supplémentaires : le plan d'expérience doit être équilibré (*i.e.* contenir autant d'individus dans chaque combinaison des trois facteurs).

Pour réaliser le test : `perm.anova(formule)`[1] où `formule` peut être (voir fiche **48** pour une explication détaillée de la construction d'une formule) :

- `reponse~fact.fixe1+fact.fixe2|fact.alea`
- `reponse~fact.fixe1*fact.fixe2|fact.alea`.

Si une *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `wilcox.paired.multcomp(reponse~facteur|fact.alea)`, où `facteur` est `fact.fixe1`, `fact.fixe2` ou `fact.fixe1:fact.fixe2` selon la *p-value* qui est significative.

67. Analyser des moyennes – relation(s) linéaire(s)

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lme4`, [2]`car`, [3]`RVAideMemoire`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **47** à **50**. ⚠

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle Linéaire (*Linear Model* ou LM), tandis qu'avec des séries appariées c'est un Modèle Linéaire Mixte (*Linear Mixed Model* ou LMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **5**)).

Remarque : lorsque toutes les variables explicatives sont des covariables, on est dans le cas particulier d'une *régression linéaire* (au sens des moindres carrés).

Construction du modèle

Pour créer le modèle :

- sans séries appariées : `modele<-lm(formule)`
- avec des séries appariées : `modele<-lmer(formule)`[1].

Voir fiche **48** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

- inclure un facteur permet de tester si les moyennes diffèrent entre les niveaux de ce facteur
- inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer
- inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de la validité du modèle

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Voir fiche **49** pour une explication détaillée de cette vérification.

Test(s)

Quel que soit le modèle, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[2] (voir fiche **50** pour une explication détaillée des hypothèses testées). Cependant, ce ne sont pas les mêmes tests qui sont réalisés selon le modèle :

- LM : la fonction réalise une ANOVA (*ANalysis Of VAriance*)
- LMM : la fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **51** pour réaliser ces comparaisons.

Récupération des coefficients associés aux covariables

Les valeurs de tous les *paramètres* du modèle sont obtenues *via* `summary(modele)`. Elles se trouvent dans le tableau `Coefficients` pour un LM, `Fixed effects` pour un LMM. Trois types de paramètres peuvent constituer ce tableau :

– la ligne `(Intercept)` donne la valeur (`Estimate`) et l'erreur standard (`Std. Error`) de l'ordonnée à l'origine du modèle. Elle est facilement interprétable s'il n'y a qu'une variable explicative et que celle-ci est une covariable, dans les autres cas elle est à oublier
– les lignes ayant le nom d'une covariable donnent la valeur (`Estimate`) et l'erreur standard (`Std. Error`) de la pente qui lie la variable à expliquer et la covariable en question
– les autres lignes, dans lesquelles on trouve des modalités de facteur(s), sont à oublier.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une moyenne nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction (seule la seconde est disponible pour les LMMs), les deux étant basées sur la fonction `predict()` :

– donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`
– créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.
Remarque : pour les LMMs, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument `REform=NA` à la fonction `predict()`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- lm(reponse~facteur*covariable)
```

On peut prédire une moyenne de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10))
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)))
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)))
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau)
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres. Deux types de moyennes peuvent être représentées :

– les moyennes brutes (*i.e.* calculées à partir des données brutes), avec leurs erreurs standards : voir fiches **14** et **40**, et l'utilisation de la fonction `tapply()`. Attention, dans le cas des LMMs, ces moyennes et erreurs standards ne tiennent pas compte du (des) facteur(s) aléatoire(s)
– les moyennes ajustées en fonction des autres variables du modèle, également avec leurs erreurs standards : voir fiche **51**.

Une fois les moyennes et erreurs standards récupérées, le diagramme peut être tracé (voir fiche **41**).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. tracer les points observés : `plot(reponse~covariable)`
2. créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[2]
3. ajouter la droite de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la droite est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la droite (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La droite s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer))`.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs droites sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des droites, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- lm(reponse~facteur*covariable)
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(reponse~covariable)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(covariable)[2]
```

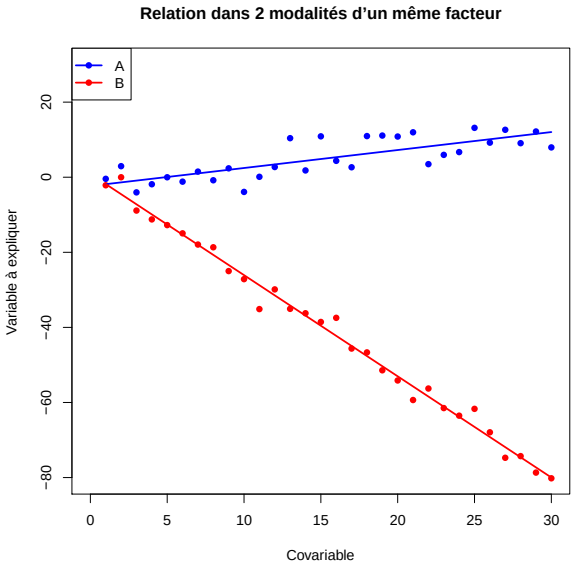
Étape 3 : ajouter la droite. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x)))))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la droite de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x)))))
```



68. Analyser des moyennes – relation non linéaire

PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE

[1]nlme, [2]RVAideMemoire

Modèle utilisé

Le modèle utilisé est un Modèle Non Linéaire (*Non Linear Model* ou NLM). Il s'utilise le plus souvent avec des variables explicatives qui sont toutes des covariables ; ce cas particulier est appelé *régression non linéaire*. Un facteur peut être ajouté pour définir des *paramètres* différents par modalité.

Construction du modèle

L'utilisation des NLM exige deux choses :

- connaître l'équation qui lie la variable à expliquer et les covariables
- avoir une idée *a priori* de la valeur des paramètres de cette équation (la majeure partie du temps, on peut se faire cette idée à partir d'un graphe).

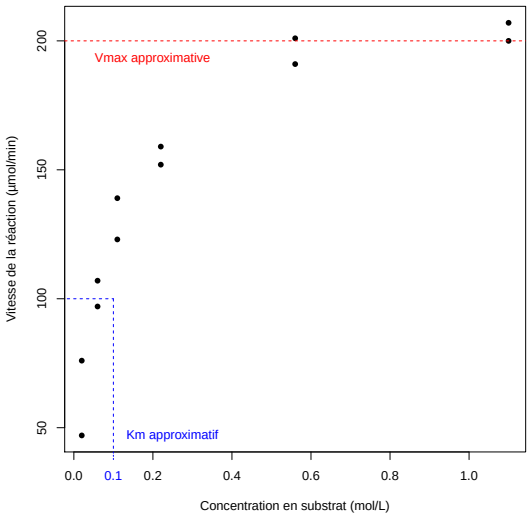
Le modèle peut ensuite être créé : `modele<-nls(reponse~équation,start=list(paramètres))`, où `équation` est l'équation explicite de la relation et `paramètres` la liste des paramètres de l'équation avec pour chacun une valeur approximative. Le symbole `~` signifie « expliqué par » ou « en fonction de ».

EXEMPLE(S)

Le modèle de Michaelis - Menten est de la forme :

$$vitesse = \frac{V_{max} \cdot concentration}{K_M + concentration}$$

On a obtenu les points expérimentaux suivants :



D'après le graphe, on estime V_{max} à 200 μmol/min et K_M à 0,1 mol/L. Le modèle s'écrit donc :

```
> modele <- nls(vitesse~Vmax*concentration/(Km+concentration),start=list(Vmax=
200,Km=0.1))
```

Certaines équations communes ont été implémentées dans R, dans des fonctions qui permettent à la fois de ne pas préciser l'équation mais aussi (et surtout) d'estimer automatiquement la valeur *a priori* des paramètres :

Type de régression	Équation	`équation`
Asymptotique		
Michaelis - Menten	$y = \frac{Vm \cdot x}{K+x}$	`SSmicmen(x,Vm,K)`
Exponentielle à 2 paramètres	$y = Asym(1 - e^{-e^{lrc}x})$	`SSasympOrig(x,Asym,lrc)`
Exponentielle à 3 paramètres	$y = Asym + (R_0 - Asym)e^{-e^{lrc}x}$	`SSasymp(x,Asym,R0,lrc)`
Sigmoïde		
Logistique à 3 paramètres	$y = \frac{Asym}{1+e^{\frac{xmid-x}{scal}}}$	`SSlogis(x,Asym,xmid,scal)`
Logistique à 4 paramètres	$y = A + \frac{B-A}{1+e^{\frac{xmid-x}{scal}}}$	`SSfpl(x,A,B,xmid,scal)`
Weibull	$y = Asym - Drop \cdot e^{-e^{lrc}x^{pwr}}$	`SSweibull(x,Asym,Drop,lrc,pwr)`
Gompertz	$y = Asym \cdot e^{-b2 \cdot b3^x}$	`SSgompertz(x,Asym,b2,b3)`
En cloche		
Biexponentielle	$y = A1 \cdot e^{-e^{lrc1}x} + A2 \cdot e^{-e^{lrc2}x}$	`SSbiexp(x,A1,lrc1,A2,lrc2)`

Utiliser ces fonctions simplifie considérablement la construction du modèle.

EXEMPLE(S)

Sur le même modèle de Michaelis - Menten, notre premier modèle :

```
> modele <- nls(vitesse~Vmax*concentration/(Km+concentration),start=list(Vmax=200,Km=0.1))
```

est équivalent à :

```
> modele <- nls(vitesse~SSmicmen(concentration,Vmax,Km))
```

Pour estimer des paramètres différents selon les modalités d'un facteur, utiliser la fonction `nlsList()`[1] à la place de `nls()` (les deux fonctions sont basées sur la même syntaxe) et ajouter `|facteur` après `equation`, où `facteur` est un vecteur contenant la modalité de chaque individu (dans le même ordre que la variable à expliquer et les covariables).

EXEMPLE(S)

Toujours sur le même modèle de Michaelis - Menten, si l'on a réalisé l'expérience avec deux enzymes et que l'on veut estimer des paramètres différents pour chacune d'entre elles :

```
> modele <- nlsList(vitesse~Vmax*concentration/(Km+concentration)|enzyme,start=list(Vmax=200,Km=0.1))
```
[1]

ou :

```
> modele <- nlsList(vitesse~SSmicmen(concentration,Vmax,Km)|enzyme)
```
[1]

Vérification de la validité du modèle

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Voir fiche **49** pour une explication détaillée de cette vérification.

Récupération des paramètres et tests

Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles se trouvent dans le tableau `Parameters` pour un modèle sans facteur (créé avec `nls()`), `Coefficients` pour un modèle avec un facteur (créé avec `nlsList()`[1]). Dans les deux cas, le tableau renvoie la valeur (`Estimate`) et l'erreur standard (`Std. Error`) de chaque paramètre, et teste leur conformité à la valeur nulle (une *p-value* non significative indique donc que le paramètre peut être supprimé de l'équation du modèle).

Prédiction à partir du modèle

L'intérêt des modèles est d'estimer les paramètres de la relation qui lie la variable à expliquer et les covariables, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une moyenne nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

- donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`
- créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.

EXEMPLE(S)

Avec notre même modèle de Michaelis - Menten et nos deux enzymes (A et B) :

```
> modele <- nlsList(vitesse~SSmicmen(concentration,Vmax,Km)|enzyme)
```
[1]

On peut prédire une valeur de vitesse de cette façon :

```
> predict(modele,newdata=list(enzyme="A",concentration=0.5))
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(enzyme=c("A","B"),concentration=c(0.5,0.5)))
```

Ou encore :

```
> predict(modele,newdata=list(enzyme=c("A","B"),concentration=rep(0.5,2)))
```

Ou encore créer un tableau de ce type :

```
> tableau
  enzyme concentration
1      A           0.5
2      B           0.5
```

Puis :

```
> predict(modele,newdata=tableau)
```

Graphes

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. tracer les points observés : `plot(reponse~covariable)`
2. créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[2]
3. ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer))`.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- nlsList(vitesse~SSmicmen(concentration,Vmax,Km)|enzyme)
```
[1]

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(vitesse~concentration)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(concentration)
```
[2]

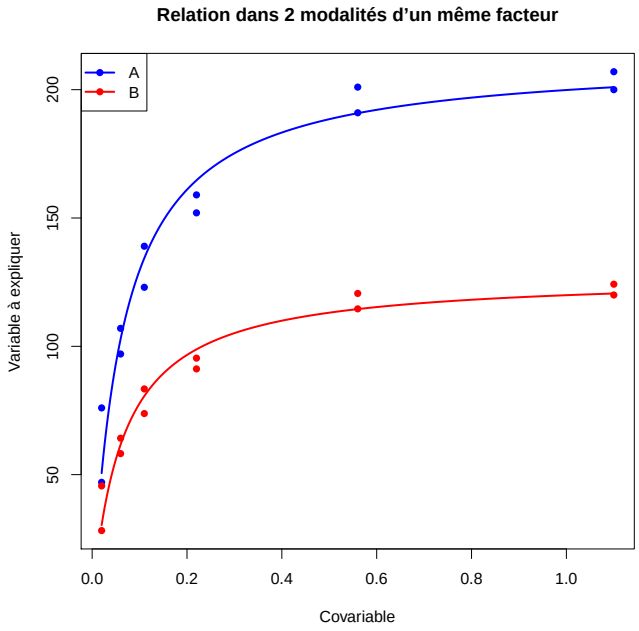
Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(concentration=x,enzyme=rep("A",length(x))
)))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(concentration=x,enzyme=rep("B",length(x))
)))
```



69. Conformité d'une note avec une valeur théorique

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Attention, les notes sont des variables piégeuses, surtout si elles sont codées numériquement (*i.e.* 1/2/3/...). On pourrait en effet penser qu'il s'agit de variables quantitatives, or il n'en est rien puisque ce sont des variables qualitatives (*i.e.* des facteurs) ordinales (voir fiche **5**). La preuve : les notes (1/2/3) peuvent parfaitement être remplacées par des lettres (A/B/C) ou des expressions (bon/moyen/mauvais) sans changer le sens de la variable. On parle aussi parfois de variables « semi-quantitatives » mais il ne faut pas se méprendre, elles n'ont rien de quantitatif. C'est pour cette raison que le paramètre de position (voir fiche **14**) le plus pertinent qui leur correspond est la *médiane*, et non la moyenne.

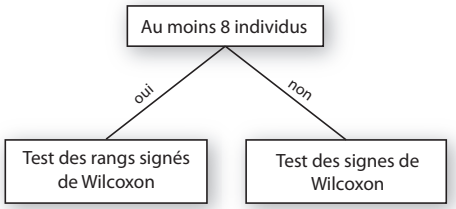
Les notes doivent être placées dans un type d'objet particulier, un *facteur ordonné*. Pour créer cet objet : `notes<-factor(serie,levels=classement,ordered=TRUE)` où `serie` est la série de données et `classement` un vecteur donnant le classement des notes dans l'*ordre croissant* (entre guillemets).

— EXEMPLE(S) —

Si les notes sont codées numériquement 1/2/3/4 (1 étant le rang le plus haut) :
```
notes <- factor(serie,levels=c("4","3","2","1"),ordered=TRUE)
```
Lorsque l'on affiche le vecteur `notes`, les niveaux sont donnés sous cette forme :
```
Levels: 4 < 3 < 2 < 1
```

Si les notes sont codées en expression bon/moyen/mauvais (bon étant le rang le plus haut) :
```
notes <- factor(serie,levels=c("mauvais","moyen","bon"),ordered=TRUE)
```
Lorsque l'on affiche le vecteur `notes`, les niveaux sont donnés sous cette forme :
```
Levels: mauvais < moyen < bon
```

Pour choisir le test approprié :



Test des rangs signés de Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.rating.test(notes,mu=n.theo)`[1] où `n.theo` est la note théorique à comparer (entre guillemets).

Ce test compare la note *médiane* de l'échantillon avec la valeur théorique. Cette note médiane est obtenue *via* `median(notes)`.

Remarque : pour pouvoir utiliser la fonction `median()` sur un facteur ordonné il est nécessaire d'avoir chargé le package `RVAideMemoire`.

Test des signes de Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.rating.signtest(notes,mu=n.theo)`[1].

Ce test compare la note *médiane* de l'échantillon avec la valeur théorique.

70. Comparaison de deux notes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Attention, les notes sont des variables piégeuses, surtout si elles sont codées numériquement (*i.e.* 1/2/3/...). On pourrait en effet penser qu'il s'agit de variables quantitatives, or il n'en est rien puisque ce sont des variables qualitatives (*i.e.* des facteurs) ordinales (voir fiche **5**). La preuve : les notes (1/2/3) peuvent parfaitement être remplacées par des lettres (A/B/C) ou des expressions (bon/moyen/mauvais) sans changer le sens de la variable. On parle aussi parfois de variables « semi-quantitatives » mais il ne faut pas se méprendre, elles n'ont rien de quantitatif. C'est pour cette raison que le paramètre de position (voir fiche **14**) le plus pertinent qui leur correspond est la *médiane*, et non la moyenne.

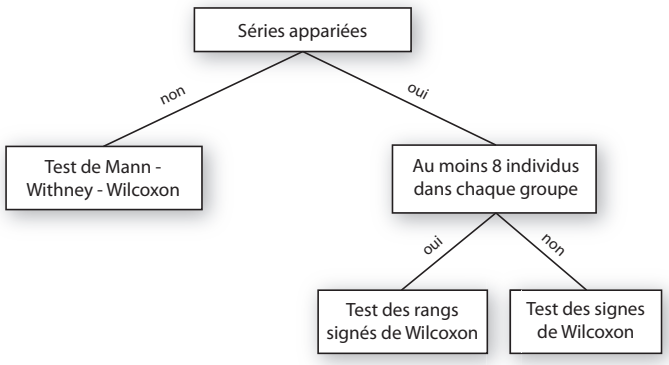
Les notes doivent être placées dans un type d'objet particulier, un *facteur ordonné*. Pour créer cet objet : `notes<-factor(serie,levels=classement,ordered=TRUE)` où `serie` est la série de données et `classement` un vecteur donnant le classement des notes dans l'*ordre croissant* (entre guillemets).

— EXEMPLE(S) —

Si les notes sont codées numériquement 1/2/3/4 (1 étant le rang le plus haut) :
`notes <- factor(serie,levels=c("4","3","2","1"),ordered=TRUE)`
Lorsque l'on affiche le vecteur `notes`, les niveaux sont donnés sous cette forme :
`Levels: 4 < 3 < 2 < 1`

Si les notes sont codées en expression bon/moyen/mauvais (bon étant le rang le plus haut) :
`notes <- factor(serie,levels=c("mauvais","moyen","bon"),ordered=TRUE)`
Lorsque l'on affiche le vecteur `notes`, les niveaux sont donnés sous cette forme :
`Levels: mauvais < moyen < bon`

Pour choisir le test approprié :



Séries non appariées

Test de Mann - Whitney - Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.rating.test(notes~facteur)`[1], où `notes` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Ce test compare la note *médiane* des deux échantillons. Pour obtenir ces notes médianes : `tapply(notes,facteur,median)`.

Remarque : pour pouvoir utiliser la fonction `median()` sur un facteur ordonné il est nécessaire d'avoir chargé le package `RVAideMemoire`.

Séries appariées

Test des rangs signés de Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.rating.test(notes~facteur,paired=TRUE)`[1].
Ce test compare la note *médiane* des deux échantillons.

Test des signes de Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.rating.signtest(notes~facteur)`[1].
Ce test compare la note *médiane* des deux échantillons.

71. Comparaison de plus de deux notes – un facteur

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Attention, les notes sont des variables piégeuses, surtout si elles sont codées numériquement (*i.e.* 1/2/3/...). On pourrait en effet penser qu'il s'agit de variables quantitatives, or il n'en est rien puisque ce sont des variables qualitatives (*i.e.* des facteurs) ordinales (voir fiche **5**). La preuve : les notes (1/2/3) peuvent parfaitement être remplacées par des lettres (A/B/C) ou des expressions (bon/moyen/mauvais) sans changer le sens de la variable. On parle aussi parfois de variables « semi-quantitatives » mais il ne faut pas se méprendre, elles n'ont rien de quantitatif. C'est pour cette raison que le paramètre de position (voir fiche **14**) le plus pertinent qui leur correspond est la *médiane*, et non la moyenne.

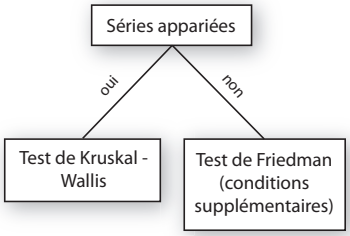
Les notes doivent être placées dans un type d'objet particulier, un *facteur ordonné*. Pour créer cet objet : `notes<-factor(serie,levels=classement,ordered=TRUE)` où `serie` est la série de données et `classement` un vecteur donnant le classement des notes dans l'*ordre croissant* (entre guillemets).

— EXEMPLE(S) —

Si les notes sont codées numériquement 1/2/3/4 (1 étant le rang le plus haut) :
`notes <- factor(serie,levels=c("4","3","2","1"),ordered=TRUE)`
Lorsque l'on affiche le vecteur `notes`, les niveaux sont donnés sous cette forme :
`Levels: 4 < 3 < 2 < 1`

Si les notes sont codées en expression bon/moyen/mauvais (bon étant le rang le plus haut) :
`notes <- factor(serie,levels=c("mauvais","moyen","bon"),ordered=TRUE)`
Lorsque l'on affiche le vecteur `notes`, les niveaux sont donnés sous cette forme :
`Levels: mauvais < moyen < bon`

Pour choisir le test approprié :



Séries non appariées

Test de Kruskal - Wallis (non paramétrique)

Pour réaliser le test : `kruskal.rating.test(notes~facteur)`[1], où `notes` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Une *p-value* significative indique qu'au moins deux classes du facteur ont un effet différent sur la variable à expliquer (sans préciser lesquelles). Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les classes en question, *via* `pairwise.wilcox.rating.test(notes,facteur)`[1].

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles moyennes sont responsables du rejet de l'hypothèse nulle dans le test global.

Ces tests comparent la note *médiane* des différents échantillons. Pour obtenir ces notes médianes : `tapply(notes,facteur,median)`.

Remarque : pour pouvoir utiliser la fonction `median()` sur un facteur ordonné il est nécessaire d'avoir chargé le package `RVAideMemoire`.

Séries appariées

Test de Friedman (non paramétrique)

Conditions supplémentaires : le plan d'expérience doit être en blocs aléatoires complets sans répétition (voir fiche **7**).

Pour réaliser le test : `friedman.rating.test(notes~fact.fixe|fact.alea)`[1] où `notes`, `fact.fixe` et `fact.alea` sont des vecteurs contenant la valeur de chaque individu (dans le même ordre) pour la variable à expliquer, le facteur (fixe) dont on veut comparer les modalités et le facteur (aléatoire) servant à définir les séries appariées, respectivement.

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `wilcox.paired.rating.multcomp(notes~fact.fixe|fact.alea)`[1].

Ces tests comparent la note *médiane* des différents échantillons.

72. Analyser des notes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`ordinal`, [2]`car`, [3]`RVAideMemoire`, [4]`lsmeans`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. ⚠
Il est indispensable de maîtriser les notions des fiches **47** à **50**.

Attention, les notes sont des variables piégeuses, surtout si elles sont codées numériquement (*i.e.* 1/2/3/...). On pourrait en effet penser qu'il s'agit de variables quantitatives, or il n'en est rien puisque ce sont des variables qualitatives (*i.e.* des facteurs) ordinales (voir fiche **5**). La preuve : les notes (1/2/3) peuvent parfaitement être remplacées par des lettres (A/B/C) ou des expressions (bon/moyen/mauvais) sans changer le sens de la variable. On parle aussi parfois de variables « semi-quantitatives » mais il ne faut pas se méprendre, elles n'ont rien de quantitatif. C'est pour cette raison que le paramètre de position (voir fiche **14**) le plus pertinent qui leur correspond est la *médiane*, et non la moyenne.

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle à Odds Proportionnels (*Proportional Odds Model* ou POM), tandis qu'avec des séries appariées c'est un Modèle à Odds Proportionnels Mixte (*Proportional Odds Mixed Model* ou POMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **5**)).

Ces modèles sont une sorte d'extension de ceux permettant d'analyser des probabilités de réponse (voir fiche **54**), mais ne rentrent pas dans la famille des Modèles Linéaires Généralisés (*Generalized Linear Models* ou GLMs). Dans le cas de notes, l'analyse consiste en fait à étudier la probabilité de passer d'une note à la note supérieure, et ce pour chaque transition du classement.

Remarque : les POM(M)s sont en fait un cas particulier des Modèles à Lien Cumulatif (*Cumulative Link (Mixed) Models* ou CLM(M)s). Le cas décrit ici – le plus fréquent – est celui d'un modèle *logistique*.

Construction du modèle

Les notes doivent être placées dans un type d'objet particulier, un *facteur ordonné*. Pour créer cet objet : `notes<-factor(serie,levels=classement,ordered=TRUE)` où `serie` est la série de données et `classement` un vecteur donnant le classement des notes dans l'*ordre croissant* (entre guillemets).

— EXEMPLE(S) —

Si les notes sont codées numériquement 1/2/3/4 (1 étant le rang le plus haut) :
```
notes <- factor(serie,levels=c("4","3","2","1"),ordered=TRUE)
```
Lorsque l'on affiche le vecteur `notes`, les niveaux sont donnés sous cette forme :
```
Levels: 4 < 3 < 2 < 1
```

Si les notes sont codées en expression bon/moyen/mauvais (bon étant le rang le plus haut) :
```
notes <- factor(serie,levels=c("mauvais","moyen","bon"),ordered=TRUE)
```
Lorsque l'on affiche le vecteur `notes`, les niveaux sont donnés sous cette forme :
```
Levels: mauvais < moyen < bon
```

Pour créer le modèle :
- sans séries appariées : `modele<-clm(formule)`[1]
- avec des séries appariées : `modele<-clmm(formule)`[1].

Voir fiche **48** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :
- inclure un facteur permet de tester si les notes diffèrent entre les niveaux de ce facteur

– inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer
– inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de la dépendance des paramètres du modèle aux données

La variable à expliquer n'étant pas quantitative, la vérification de la validité du modèle ne se fait pas de manière habituelle ici. On cherche en fait à savoir à quel point les paramètres du modèle dépendent des données qu'on lui fournit. Si les paramètres sont très sensibles à la moindre variation dans les données, c'est que le modèle est mal défini. Cela peut indiquer (entre autres) que le modèle peut être simplifié ou que certains paramètres ne sont pas calculables.

La valeur qui représente cette dépendance est le *conditionnement* de la matrice hessienne. Elle est obtenue simplement en appelant `modele`, sous l'intitulé `cond.H`. Il n'y a pas vraiment de seuil absolu, mais on considère généralement que si elle est supérieure à 10^4 c'est que le modèle est mal défini.

Test(s)

Quel que soit le modèle, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[2] (voir fiche **50** pour une explication détaillée des hypothèses testées). Le test réalisé est un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Remarque : pour pouvoir utiliser la fonction `Anova()`[2] sur un modèle créé par `clm()`[1] ou `clmm()`[1] il est nécessaire d'avoir chargé le package `RVAideMemoire`.

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui différent. Voir fiche **51** pour réaliser ces comparaisons.

Prédiction à partir du modèle (non disponible pour les POMMs)

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une note nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

– donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`
– créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.

La spécificité des POM(M)s (et plus globalement des CLM(M)s) est qu'ils travaillent sur plusieurs valeurs simultanément, *i.e.* les probabilités associées *à chaque note* du classement. La fonction `predict()` renvoie donc, pour chaque prédiction, autant de valeurs qu'il y a de notes possibles. Ces valeurs correspondent à la probabilité respective de chaque note, sachant la valeur des variables explicatives qui ont servi à la prédiction. Les prédictions sont organisées en une matrice (qui est en fait dans le compartiment `fit` de la liste renvoyée par `predict()`) où chaque ligne est une prédiction et chaque colonne une note du classement (du bas vers le haut du classement).

Deux autres types de prédiction sont disponibles :

– les probabilités cumulées, en ajoutant l'argument `type="cum.prob"` à la fonction `predict()`. La fonction renvoie également une matrice (dans le compartiment `cprob1` d'une liste) avec une ligne par prédiction et une colonne par note. Chaque colonne contient non pas la probabilité de la note

correspondante, mais la probabilité que la note obtenue soit *inférieure ou égale* à cette note. La dernière colonne (*i.e.* la note la plus élevée du classement) contient donc nécessairement des 1

– la note la plus probable, en ajoutant l'argument `type="class"` à la fonction `predict()`. La fonction renvoie cette fois un vecteur (dans le compartiment `fit` d'une liste) contenant une valeur par prédiction. Cette valeur est la note la plus probable sachant la valeur des variables explicatives qui ont servi à la prédiction.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- clm(reponse~facteur*covariable)[1]
```

On peut prédire une note de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10))
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)))
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)))
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau)
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres (voir fiche **41**). Mais du fait de la complexité d'une variable à expliquer telle que des notes, plusieurs représentations sont possibles. On a également le choix de l'information que l'on veut représenter : la fréquence observée des différentes notes (dans le jeu de données) ou la probabilité ajustée de ces notes (calculée à partir du modèle). Il est souvent plus pertinent de représenter les probabilités ajustées, car elles illustrent la variation réellement due au facteur, après avoir retiré la variation dues aux autres variables explicatives du modèles. De plus, seules les probabilités ajustées prennent en compte l'effet des facteurs aléatoires (s'il y en a dans l'analyse).

Les représentations possibles sont :

– la fréquence / probabilité de chaque note.
Les fréquences observées sont obtenues *via* `rating.prob(notes,facteur)`[3] (`facteur` peut être une interaction entre deux facteurs, spécifiée par `facteur1:facteur2`).
Deux étapes sont nécessaires pour récupérer les probabilités ajustées. D'abord calculer ces probabilités et les stocker dans un objet (appelé `LSM` ici) : `LSM <- lsmeans(modele,~facteur|cut)`[4] (voir fiche **51** pour une explication de la syntaxe de la fonction `lsmeans()`[4] ; il est indispensable que la formule se termine par `|cut`). Ensuite récupérer les probabilités, qui ne sont pas données directement par la fonction `lsmeans()`[4] : `rating.lsmeans(LSM)`[3]

– la fréquence cumulée / probabilité cumulée de chaque note (*i.e.* la probabilité que la note soit *inférieure ou égale* à chaque note possible, du bas vers le haut du classement).
Les fréquences cumulées observées sont obtenues *via* `rating.prob(notes,facteur,type="cumprob")`[3].
Les probabilités ajustées sont récupérées (à partir du même objet `LSM`) *via* `rating.lsmeans(LSM,type="cumprob")`[3]

– la note la plus fréquente / probable. Le diagramme en barres ne peut pas être utilisé dans ce cas car la valeur est qualitative et non quantitative.
La note la plus fréquente est obtenue *via* `rating.prob(notes,facteur,type="class")`[3].
La note la plus probable est obtenue (à partir du même objet `LSM`) *via* `rating.lsmeans(notes,facteur,type="class1")`[3].

73. Analyser des temps de survie – Modèle Linéaire Généralisé

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lme4`, [2]`car`, [3]`survival`, [4]`RVAideMemoire`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **47** à **50**. ⚠

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM), tandis qu'avec des séries appariées c'est un Modèle Linéaire Généralisé Mixte (*Generalized Linear Mixed Model* ou GLMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **5**)).

Contrairement au Modèle Linéaire (et sa variante Mixte) utilisé pour analyser des moyennes, les GLM(M)s ne sont pas basés sur une loi normale. Dans le cas de temps de survie, la loi à utiliser est une loi Gamma.

Construction du modèle

Pour créer le modèle :

- sans séries appariées : `modele<-glm(formule,family="Gamma")`
- avec des séries appariées : `modele<-glmer(formule,family="Gamma")`[1].

Voir fiche **48** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

- inclure un facteur permet de tester si les temps de survie diffèrent entre les niveaux de ce facteur
- inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer
- inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Voir fiche **49** pour une explication détaillée de cette vérification.

Test(s)

Quel que soit le modèle, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[2] (voir fiche **50** pour une explication détaillée des hypothèses testées). Cependant, ce ne sont pas les mêmes tests qui sont réalisés selon le modèle :

- GLM : la fonction réalise un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé)
- GLMM : la fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **51** pour réaliser ces comparaisons.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire un temps de survie nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction (seule la seconde est disponible pour les GLMMs), les deux étant basées sur la fonction `predict()` :

- donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables),type="response")`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`
- créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau,type="response")`.
  Remarque : pour les GLMMs, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument `REform=NA` à la fonction `predict()`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- glm(reponse~facteur*covariable,family="Gamma")
```

On peut prédire un temps de survie de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10),type="response")
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)),type="response")
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)),type="response")
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau,type="response")
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement des courbes de survie (dites de Kaplan - Meier). Leur tracé nécessite trois étapes :

1. créer un *objet de survie*, une forme particulière des temps de survie : `survie<-Surv(reponse)`[3] où `reponse` est la variable à expliquer du modèle, *i.e.* les temps de survie
2. créer les données du graphe : `courbes<-survfit(survie~facteur)`[3]
3. tracer le graphe : `plot(courbes)`. Il est possible d'ajouter l'intervalle de confiance (à 95 %) de chaque courbe en ajoutant l'argument `conf.int=TRUE` (attention cependant à ne pas multiplier les courbes, qui rendent le graphe difficilement lisible). Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. tracer les points observés : `plot(reponse~covariable)`
2. créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[4]
3. ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer,type="response"))`.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- glm(reponse~facteur*covariable,family="Gamma")
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(reponse~covariable)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(covariable)
```
[3]

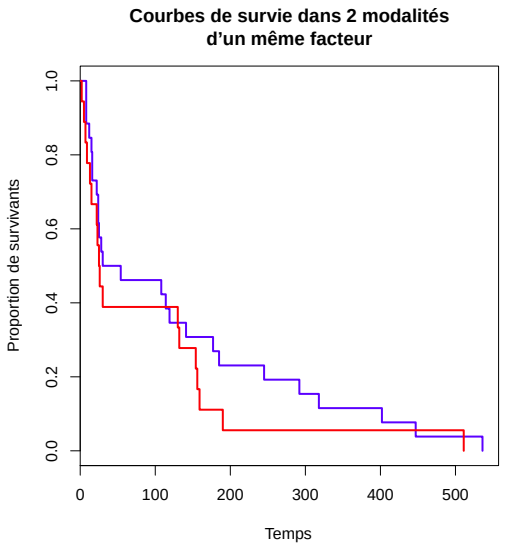
Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

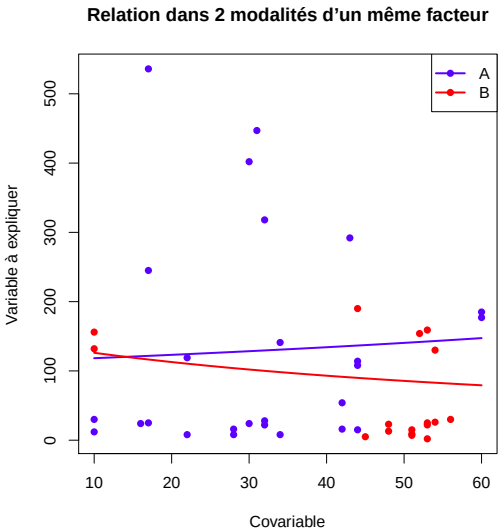
```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x))),
type="response"))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x))),
type="response"))
```





74. Analyser des temps de survie – Régression de survie

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`survival`, [2]`car`, [3]`RVAideMemoire`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **47** à **50**. ⚠

Construction du modèle

Dans une régression de survie, la variable à expliquer n'est pas directement le temps avant la mort des individus, mais un objet particulier appelé *objet de survie*. Cet objet prend en compte à la fois le temps de survie, mais aussi des possibles censures (*i.e.* le fait que la mort ne soit pas observée avant la fin de l'étude ; voir le sommaire pour plus d'explications). Pour le créer :

- `survie<-Surv(mort)`[1] s'il n'y a aucun individu censuré, où `mort` est le vecteur contenant les temps de survie
- `survie<-Surv(mort,censure)`[1] s'il y a des individus censurés, où `mort` est le vecteur contenant les temps de survie et `censure` le vecteur contenant l'indication de censure de chaque individu (0 : censure, 1 : pas de censure *i.e.* mort observée).

Dans la formule du modèle, la variable à expliquer est `survie`, l'objet de survie.

Pour ensuite créer le modèle :

- si le risque instantané est constant (voir le sommaire pour tester cette hypothèse) : `modele<-survreg(formule,dist="exponential")`[1]. La loi exponentielle sert à modéliser la constance du risque instantané
- si le risque instantané n'est pas constant : `modele<-survreg(formule,dist="weibull")`[1]. La loi de Weibull sert à modéliser l'évolution du risque instantané au cours du temps (qu'il diminue ou qu'il augmente).

Voir fiche **48** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

- inclure un facteur permet de tester si les temps de survie diffèrent entre les niveaux de ce facteur
- inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer
- inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Dans le cas d'un modèle avec risque instantané non constant, l'évolution de ce risque est donnée par `summary(modele)`. Si le paramètre `Scale` est < 1 le risque diminue avec le temps, s'il est > 1 il augmente.

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Voir fiche **49** pour une explication détaillée de cette vérification.

Test(s)

Quel que soit le modèle, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[2] (voir fiche **50** pour une explication détaillée des hypothèses testées). Le test réalisé est un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **51** pour réaliser ces comparaisons.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire un temps de survie nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

- donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`
- créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- survreg(survie~facteur*covariable,dist="exponential")
```
[1]

On peut prédire un temps de survie de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10))
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)))
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)))
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau)
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement des courbes de survie (dites de Kaplan - Meier). Leur tracé nécessite deux étapes :

1. créer les données du graphe : `courbes<-survfit(survie~facteur)`[1]
2. tracer le graphe : `plot(courbes)`. Si des individus sont censurés, ils sont représentés par des croix (+). Il est possible d'ajouter l'intervalle de confiance (à 95 %) de chaque courbe en ajoutant l'argument `conf.int=TRUE` (attention cependant à ne pas multiplier les courbes, qui rendent le graphe difficilement lisible). Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. tracer les points observés : `plot(mort~covariable)`
2. créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[3]

3. ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer))`.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- survreg(survie~facteur*covariable,dist="exponential")
```
[1]

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(mort~covariable)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(covariable)
```
[3]

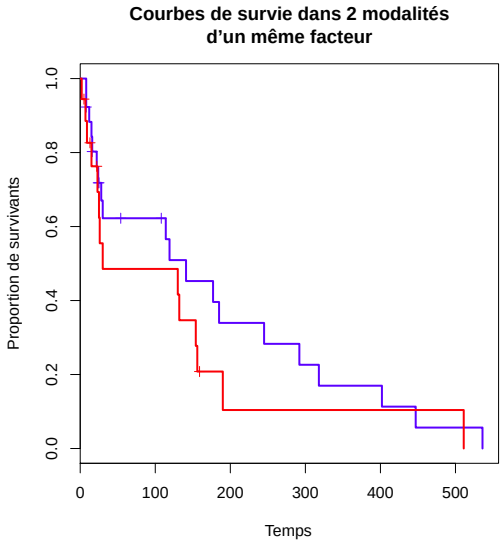
Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

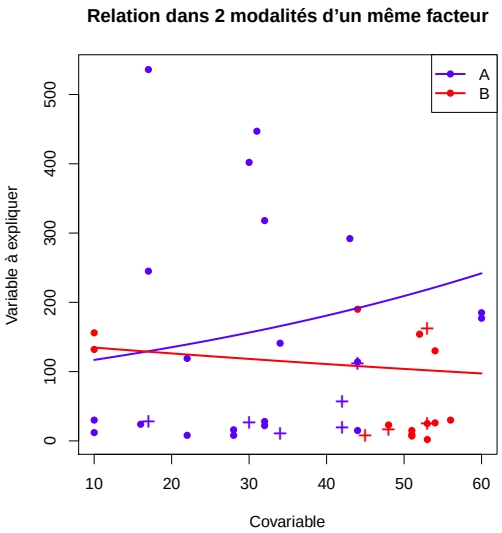
```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x)))
))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x)))
))
```





75. Analyser des temps de survie – Modèle de Cox

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`survival`, [2]`RVAideMemoire`, [3]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **47** à **50**. ⚠

Construction du modèle

Dans un modèle de Cox, la variable à expliquer n'est pas directement le temps avant la mort des individus, mais un objet particulier appelé *objet de survie*. Cet objet prend en compte à la fois le temps de survie, mais aussi des possibles censures (*i.e.* le fait que la mort ne soit pas observée avant la fin de l'étude ; voir le sommaire pour plus d'explications). Pour le créer :

- `survie<-Surv(mort)`[1] s'il n'y a aucun individu censuré, où `mort` est le vecteur contenant les temps de survie
- `survie<-Surv(mort,censure)`[1] s'il y a des individus censurés, où `mort` est le vecteur contenant les temps de survie et `censure` le vecteur contenant l'indication de censure de chaque individu (0 : censure, 1 : pas de censure *i.e.* mort observée).

Dans la formule du modèle, la variable à expliquer est `survie`, l'objet de survie.

Pour ensuite créer le modèle : `modele<-coxph(formule)`[1]. Voir fiche **48** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

- inclure un facteur permet de tester si les temps de survie diffèrent entre les niveaux de ce facteur
- inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer
- inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Le modèle de Cox est un peu particulier puisque, contrairement aux autres modèles d'analyse de temps de survie, il n'est pas paramétrique mais semi-paramétrique (ce qui lié au fait qu'on ne puisse pas l'utiliser dans un but prédictif).

Vérification de la validité du modèle

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Le modèle de Cox a des conditions de validité particulières :

- la relation entre chaque variable explicative quantitative (ou covariable) et le risque instantané doit être log-linéaire. Pour tester cette hypothèse : `cox.resid(modele)`[2]. La fonction trace un graphe par covariable, sur lequel la ligne rouge représente la tendance du nuage de point. On accepte l'hypothèse de log-linéarité pour une covariable si la ligne rouge correspondante est à peu près horizontale. Dans le cas contraire il vaut mieux la découper en classes, puis la réintégrer dans le modèle comme facteur

— EXEMPLE(S) —

Avec un modèle contenant un facteur et une covariable :

```
> modele <- coxph(survie~facteur+covariable)[1]
```

L'hypothèse de log-linéarité entre la variable à expliquer et la covariable est testée par :

```
> cox.resid(modele)[2]
```

Si l'hypothèse n'est pas respectée, on découpe la covariable en classes. Un exemple avec un découpage en deux classes :

```
> covar.class <- cut(covariable,breaks=2)
```

Remarque : pour ajouter un nom aux classes, utiliser l'argument `label`. Les classes ont par défaut la même longueur. Pour plus d'informations (beaucoup d'options sont disponibles), voir `?cut`.

Un nouveau modèle est créé avec la covariable transformée en facteur :
`> modele <- coxph(survie~facteur+covar.class)`[1]

– le rapport des risques instantanés de deux individus doit être constant au cours du temps (hypothèse des « risques proportionnels »). Pour tester cette hypothèse : `cox.zph(modele)`[1]. La fonction teste l'hypothèse pour chaque variable explicative, ainsi que pour le modèle global. Si une *p-value* est significative, cela indique que l'hypothèse n'est pas respectée pour la variable explicative en question, qui est dite *dépendante du temps*. Il vaut mieux alors l'intégrer dans le modèle en temps que *strate* et non variable explicative (pour les variables explicatives quantitatives, cela passe par un découpage en classes et une transformation en facteur). L'effet de la variable ne sera plus calculé, mais pris en compte à travers la définition de risques instantanés *de base* différents selon les strates. Pour intégrer une strate dans la formule du modèle, ajouter `+strata(variable)` après les variables explicatives (et retirer la variable désormais stratifiée des variables explicatives).

EXEMPLE(S)

Toujours avec le modèle suivant :
`> modele <- coxph(survie~facteur+covariable)`[1]
L'hypothèse des risques proportionnels est testée par :
`> cox.zph(modele)`[1]
Si l'hypothèse n'est pas respectée pour le facteur, le modèle est recréé en traitant celui-ci comme une strate :
`> modele <- coxph(survie~covariable+strata(facteur))`[1]
Si l'hypothèse n'est pas respectée pour la covariable, celle-ci doit être factorisée puis intégrée comme strate dans le modèle :
`> modele <- coxph(survie~facteur+strata(covar.class))`[1]

Test(s)

Selon que le modèle contienne une strate ou non, un test différent est à réaliser :

– modèle sans strate : `Anova(modele)`[3]. La fonction réalise un test du rapport des vraisemblances partielles (*Partial Likelihood Ratio Test* ou PLR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé)

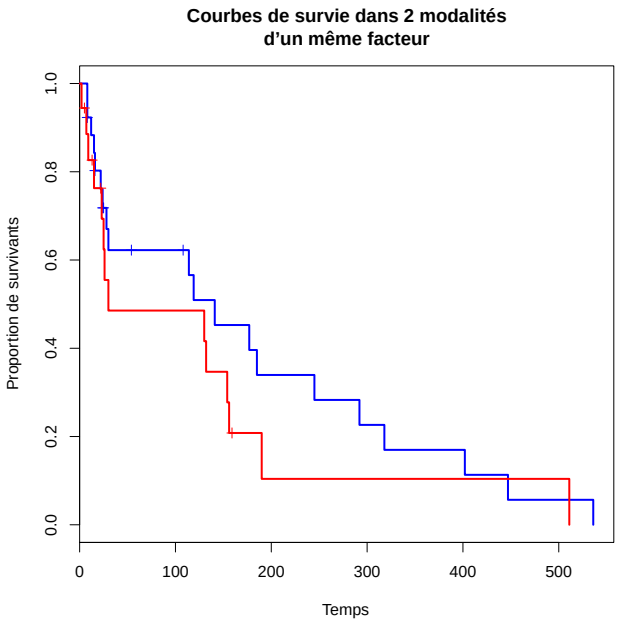
– modèle avec strate : `Anova(modele,test="Wald")`[3]. La fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui différent. Voir fiche **51** pour réaliser ces comparaisons.

Graphes

Le modèle de Cox n'étant pas prédictif, on illustre en pratique uniquement l'effet de facteurs sur le temps de survie. Pour ce faire, on réalise généralement des courbes de survie (dites de Kaplan - Meier). Leur tracé nécessite deux étapes :

1. créer les données du graphe : `courbes<-survfit(survie~facteur)`[1]. Il est possible d'ajouter une strate dans la formule (sous la forme `+strata(facteur2)`) pour tracer une courbe par modalité du facteur et par strate. Attention cependant à ne pas multiplier les courbes, qui rendent le graphe difficilement lisible
2. tracer le graphe : `plot(courbes)`. Si des individus sont censurés, ils sont représentés par des croix (+). Il est possible d'ajouter l'intervalle de confiance (à 95 %) de chaque courbe en ajoutant l'argument `conf.int=TRUE` (même avertissement sur le nombre de courbes). Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

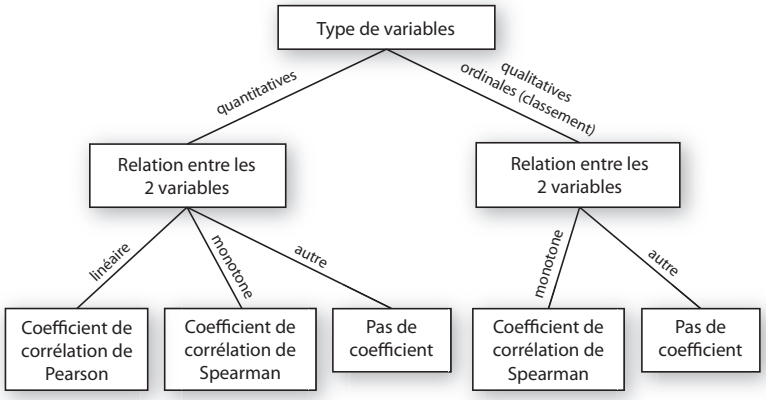

Courbes de survie dans 2 modalités
d'un même facteur
Proportion de survivants
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500
Temps

76. Intensité de la liaison entre deux variables quantitatives

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Pour vérifier le caractère linéaire ou simplement monotone (*i.e.* ascendant ou descendant) de la relation entre les deux variables, utiliser un graphe de type nuage de points (voir fiche **15**).

Lorsque l'on s'intéresse à un coefficient de corrélation, on veut généralement avoir trois informations : (i) sa valeur, (ii) son intervalle de confiance et (iii) savoir s'il est significativement différent de 0 (*i.e.* s'il y a « vraiment »une corrélation entre les deux variables).

Les coefficients de corrélation varient entre -1 (corrélation extrêmement forte et négative) et 1 (corrélation extrêmement forte et positive).

Coefficient de corrélation linéaire de Pearson (paramétrique)

Les trois informations sont données par la fonction `cor.test(variable1,variable2,method="pearson")`, où `variable1` et `variable2` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre). La fonction renvoie, dans l'ordre : le test de significativité (*i.e.* de comparaison par rapport à 0), l'intervalle de confiance à 95 % et la valeur du coefficient.

Coefficient de corrélation de Spearman (non paramétrique)

La valeur du coefficient de corrélation et le résultat du test de significativité sont donnés par la fonction `cor.test(variable1,variable2,method="spearman")`.

Pour obtenir l'intervalle de confiance à 95 % du coefficient, utiliser `spearman.ci(variable1,variable2)`[1]. L'intervalle de confiance est calculé par bootstrap.

77. Conformité d'un coefficient de corrélation linéaire avec une valeur théorique

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Dans la pratique, un coefficient de corrélation est le plus souvent comparé à la valeur nulle, ce qui permet de conclure s'il y a corrélation ou pas entre les deux variables.

La fonction `cor.test()` réalise systématiquement ce test et en revoie la *p-value* lorsqu'elle calcule un coefficient de corrélation (voir fiche **76**).

Pour comparer un coefficient de corrélation de Pearson à une valeur quelconque, utiliser `cor.conf(variable1,variable2,theo=valeur)`[1] où `variable1` et `variable2` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre), et `valeur` la valeur théorique à comparer (entre -1 et 1). Les conditions d'utilisation de ce test sont identiques à celles du calcul du coefficient de corrélation linéaire de Pearson, à savoir : échantillonnage aléatoire et simple et présence de deux variables quantitatives liées de façon linéaire (voir fiche **15** pour observer la liaison entre les deux variables).

De façon plus générale, on peut tester la conformité d'un coefficient de corrélation avec une valeur théorique quelconque simplement en regardant si celle-ci est contenue dans l'intervalle de confiance du coefficient (le niveau de précision de cet intervalle étant le plus souvent de 95 %).

78. Comparaison de plusieurs coefficients de corrélation linéaire

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Ces tests ne s'appliquent qu'aux coefficients de corrélation linéaire de Pearson (voir fiche **76**).

Leurs conditions d'utilisation sont identiques à celles du calcul du coefficient de corrélation linéaire de Pearson, à savoir : échantillonnage aléatoire et simple et présence de deux variables quantitatives liées de façon linéaire (voir fiche **15** pour observer la liaison entre les deux variables).

Comparaison de deux coefficients

Pour réaliser le test, utiliser selon la situation l'une des deux fonctions suivantes :

- `cor.2comp(variable1,variable2,variable3,variable4)`[1] où `variable1` et `variable2` sont des vecteurs contenant la valeur de chaque individu pour les deux variables définissant la 1ère corrélation (dans le même ordre), tandis que `variable3` et `variable4` sont des vecteurs contenant la valeur de chaque individu pour les deux variables définissant la 2nde corrélation (dans le même ordre)
- `cor.multcomp(variable1,variable2,facteur)`[1] où `variable1` et `variable2` sont des vecteurs contenant la valeur de chaque individu pour les deux variables à tester (dans le même ordre), et `facteur` un vecteur contenant la valeur de chaque individu pour le facteur définissant les deux groupes à comparer.

Si les deux coefficients de corrélation ne sont significativement pas différents, les deux fonctions renvoient la valeur du coefficient de corrélation commun, son intervalle de confiance à 95 % et le résultat du test de conformité de ce coefficient avec la valeur nulle (cette valeur théorique peut être modifiée grâce à l'argument `theo=valeur`).

Comparaison de plus de deux coefficients

Pour réaliser le test : `cor.multcomp(variable1,variable2,facteur)`[1] où `facteur` définit cette fois plus de deux groupes.

Si les coefficients de corrélation ne sont significativement pas différents, la fonction renvoie la valeur du coefficient de corrélation commun, son intervalle de confiance à 95 % et le résultat du test de conformité de ce coefficient avec la valeur nulle (cette valeur théorique peut être modifiée grâce à l'argument `theo=valeur`).

Si la *p-value* du test est significative, cela indique qu'au moins deux coefficients diffèrent l'un de l'autre, sans préciser lesquels. La fonction effectue alors toutes les comparaisons deux-à-deux possibles.

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quels coefficients sont responsables du rejet de l'hypothèse nulle dans le test global.

79. La régression linéaire simple au sens des moindres rectangles

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Modèle utilisé

Le modèle utilisé est un cas particulier de la régression linéaire classique, basée sur la méthode des *moindres carrés* (voir fiche **67**). Dans la régression linéaire au sens des moindres carrés, une variable est considérée comme à expliquer, tandis que l'autre est considérée comme explicative. Dans la régression linéaire au sens des *moindres rectangles*, les deux variables sont considérées sur un même pied d'égalité, aucune n'expliquant l'autre. On dit de ces variables qu'elles sont *interdépendantes*.

Un facteur peut être ajouté à la régression pour définir des *paramètres* (*i.e.* pente – ou coefficient directeur – et ordonnée à l'origine) différents par modalité.

Construction du modèle

Pour créer le modèle : `regression<-least.rect(variable.y~variable.x)`[1], où `variable.y` et `variable.x` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre). Les noms `variable.y` et `variable.x` n'ont qu'une valeur graphique : `variable.x` est destiné à être tracé en abscisses et `variable.y` en ordonnées.

Pour estimer des paramètres différents selon les modalités d'un facteur : `regression<-least.rect(variable.y~variable.x|facteur)`[1], où `facteur` est un vecteur contenant la modalité de chaque individu (dans le même ordre que les deux autres variables).

Vérification de la validité du modèle

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Voir fiche **49** pour une explication détaillée de cette vérification.

Récupération des paramètres et tests

Les valeurs des paramètres de la (ou les) régression(s) sont obtenues *via* `summary(regression)`, dans le tableau `Coefficients`. La fonction renvoie la valeur et l'intervalle de confiance de l'ordonnée à l'origine (`(Intercept)`) et de la pente (qui porte le nom de la variable x). Si un facteur a été ajouté à la régression, un tableau est renvoyé pour chacun de ces deux paramètres, contenant une ligne par modalité.

Dans le cadre de la régression linéaire au sens des moindres rectangles, on compare le plus souvent la pente à la valeur théorique 1 (qui correspond en allométrie à une relation d'isométrie entre les deux organes/structures comparés). Le résultat de ce test est également obtenu *via* `summary(regression)`. Pour changer la valeur théorique, ajouter l'argument `theo=valeur` à la fonction `least.rect()`[1]. Si un facteur a été défini dans le modèle, un test est réalisé pour chaque pente.

Il n'existe pas de test pour comparer plusieurs pentes ou ordonnées à l'origine. Cependant, lorsqu'un facteur a été défini dans le modèle, si les intervalles de confiance des pentes (ou des ordonnées à l'origine) ne se chevauchent pas entre deux modalités, alors ces pentes sont significativement différentes (et vice-versa).

Prédiction à partir du modèle

L'intérêt du modèle est d'estimer les paramètres de la relation qui lie les variables x et y, mais également de prédire la valeur que prendrait la variable y pour des valeurs *connues* de la variable x (ou inversement, les deux variables étant interdépendantes). Prédire une valeur de y nécessite donc de fixer la valeur de x – et du facteur s'il y en a un dans le modèle.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

– donner la valeur (ou les valeurs pour faire plusieurs prédictions) de la variable x – et éventuellement du facteur – directement dans la fonction, sous la forme d'une liste : `predict(modele,ne-`

wdata=list(variables))`, où `variables` est un enchaînement de `variable.x=valeur,facteur=valeur`

– créer un tableau contenant une colonne par chacune des deux variables connues (x et éventuellement facteur ; les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.

EXEMPLE(S)

Avec un modèle contenant une variable x variant de 0 à 30 et un facteur à deux niveaux (A et B) :

```
> regression <- least.rect(variable.y~variable.x|facteur)[1]
```

On peut prédire une valeur de y de cette façon :

```
> predict(regression,newdata=list(variable.x=10,facteur="A"))
```

Ou, pour plusieurs prédictions :

```
> predict(regression,newdata=list(variable.x=c(10,10),facteur=c("A","B")))
```

Ou encore :

```
> predict(regression,newdata=list(variable.x=rep(10,2),facteur=c("A","B")))
```

Ou encore créer un tableau de ce type :

```
> tableau
  variable.x facteur
1         10        A
2         10        B
```

Puis :

```
> predict(regression,newdata=tableau)
```

Graphes

Illustrer la relation entre les variables x et y nécessite trois étapes :

1. tracer les points observés : `plot(variable.y~variable.x)`
2. créer un vecteur ayant les mêmes minimum et maximum que la variable x mais découpé en très petits intervalles : `x <- seq2(variable.x)`[1]
3. ajouter la droite de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la droite est en fait basée sur une prédiction : la valeur que prend la variable y pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la droite (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient un facteur, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la variable x change de valeur pour toutes les prédictions). La droite s'ajoute *via* `lines(x,predict(regression,newdata=variables))` où `variables` correspond aux valeurs de la variable x et du facteur.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs droites sur le même graphe, par exemple pour plusieurs niveaux du facteur.

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> regression <- least.rect(variable.y~variable.x|facteur)[1]
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(variable.y~variable.x)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(variable.x)[1]
```

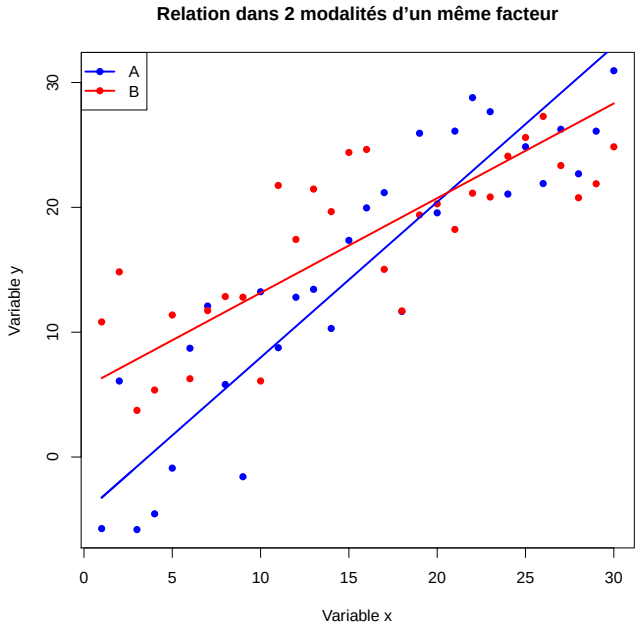
Étape 3 : ajouter la droite. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(regression,newdata=list(variable.x=x,facteur=rep("A",length(
x)))))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la droite de la relation dans la modalité B :

```
> lines(x,predict(regression,newdata=list(variable.x=x,facteur=rep("B",length(x)
))))
```

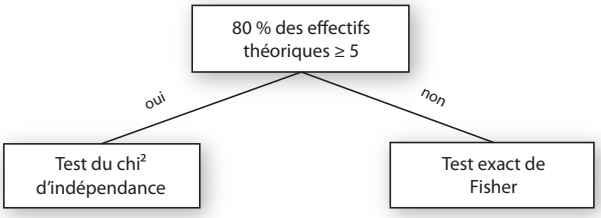


80. Association entre deux variables qualitatives

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Les données doivent être organisées en un tableau de contingence du type :

		Variable B		
		Classe 1	…	Classe c
Variable A	Classe 1			
	…			
	Classe k			

où chaque case contient le nombre d'individus possédant à la fois le caractère de la variable A et celui de la variable B.

Ce tableau est obtenu de la manière suivante : `tab.cont<-table(variableA,variableB)` où `variableA` et `variableB` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre).

Les effectifs théoriques sont donnés par la fonction `chisq.test(tab.cont)$expected`. Ils permettent de vérifier si 80 % d'entre eux sont ≥ 5. Cette règle est appelée « règle de Cochran ».

Test du χ^2 d'indépendance (ou d'homogénéité ; non paramétrique)

Pour réaliser le test : `chisq.test(tab.cont)`.

Remarque : la fonction `chisq.test()` n'utilise pas la règle de Cochran et renvoie un avertissement dès qu'au moins un effectif théorique est < 5. Si la règle est bien respectée, ne pas tenir compte de l'avertissement.

Une *p-value* significative indique que les deux variables ne sont pas indépendantes, sans préciser les classes qui sont à l'origine de cette liaison. Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les classes en question, *via* `fisher.multcomp(tab.cont)`[1]. La fonction réalise un test exact de Fisher sur chaque tableau de contingence 2 x 2 possible à partir de `tab.cont`. Il est nécessaire d'interpréter ces résultats pour repérer les classes qui apparaissent systématiquement dans les tests qui donnent une *p-value* significative. Ce sont ces classes qui sont liées.

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune liaison significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles classes sont responsables du rejet de l'hypothèse nulle dans le test global.

EXEMPLE(S)

Les caractères étudiés sont la couleur des cheveux et des yeux de 116 individus :

```
> tab.cont
        bleu  marron  vert
 blond    25       6     9
 brun     10      15    16
 roux     12      14     9
> fisher.multcomp(tab.cont)[1]
      Pairwise comparisons by Fisher's exact test for count data

data: tab.cont

             bleu:marron  bleu:vert  marron:vert
 blond:brun      0.02211    0.03435       0.7793
 blond:roux      0.03435    0.44655       0.4801
 brun:roux       0.77935    0.44655       0.5362

P value adjustment method: fdr
```

Les classes qui sont liées sont ici `blond` et `bleu`.

Si la règle de Cochran est respectée et que chaque classe des deux variables contient au moins 5 % du nombre total d'individus, il est possible de calculer un coefficient mesurant l'association entre les deux variables : le coefficient d'association de Cramer (non paramétrique). Pour ce faire : `cramer.test(tab.cont)`[1]. La fonction renvoie également l'intervalle de confiance du coefficient (calculé par bootstrap) et le résultat du test de significativité (*i.e.* de comparaison à la valeur nulle ; ce test est strictement le même que le χ^2 d'indépendance). Dès qu'au moins l'une des deux variables possède plus de deux classes, le coefficient varie entre 0 et 1. Dans le cas de deux facteurs à chacun deux classes, il varie comme les coefficients de corrélation entre -1 et 1.

Test exact de Fisher (non paramétrique)

Pour réaliser le test : `fisher.test(tab.cont)`.

Si le message d'avertissement `out of workspace` apparaît, augmenter la valeur de l'argument `workspace` (par défaut `workspace=200000`). Si un autre message d'avertissement apparaît, cela peut être à cause d'un tableau trop complexe à analyser.

Si la *p-value* du test est significative, les comparaisons multiples se font également *via* `fisher.multcomp(tab.cont)`[1].

81. Analyser des moyennes multivariées

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire, [2]car

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **47** à **50**. ⚠

Modèle utilisé

Le modèle utilisé est un Modèle Linéaire Multivarié (*Multivariate Linear Model* ou MLM), une extension du Modèle Linéaire classique utilisé pour analyser des moyennes univariées (voir fiche **67**). À la différence du Modèle Linéaire classique, il y a plusieurs variables à expliquer dans le MLM.

Vérification préalable

Le MLM fait l'hypothèse que les relations entre les variables à expliquer sont linéaires. La vérification de cette hypothèse se fait en deux étapes :

1. créer l'objet contenant les variables à expliquer, qui doit être une matrice où chaque colonne correspond à une variable. Deux cas sont possibles :
   - les données sont contenues dans un tableau (par exemple importé depuis un fichier externe) : `reponse<-as.matrix(tableau[,colonnes])` où `colonnes` est un vecteur contenant le numéro (ou le nom) des colonnes du tableau correspondant aux variables à expliquer (voir fiche **2**)
   - les données sont des vecteurs, contenant chacun une variable à expliquer : `reponse<-as.matrix(data.frame(variables))` où `variables` est l'enchaînement du nom des vecteurs, séparés par une virgule
2. représenter sur un graphe toutes les relations deux-à-deux entre les variables à expliquer : `pairs(reponse,panel=panel.smooth)`. Les courbes rouges indiquent la tendance des nuages de points. Si elles sont à peu près droites, les relations peuvent être considérées comme linéaires.

— EXEMPLE(S) —

On a trois variables à expliquer. Si elles correspondent aux trois premières colonnes d'un tableau :

```
> reponse <- as.matrix(tableau[,1:3])
```

Si elles correspondent à trois vecteurs différents :

```
> reponse <- as.matrix(data.frame(vecteur1,vecteur2,vecteur3))
```

Pour ensuite représenter les relations entre ces trois variables :

```
> pairs(reponse,panel=panel.smooth)
```

Cet objet `reponse` sera la variable à expliquer du modèle.

Construction du modèle

Pour créer le modèle : `modele<-manova(formule)`. Voir fiche **48** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

- inclure un facteur permet de tester si les moyennes diffèrent entre les niveaux de ce facteur
- inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer
- inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et les variables à expliquer est différente selon la modalité du facteur.

Vérification de la validité du modèle

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). L'ajustement d'un MLM est cependant plus difficile à vérifier que celui d'un modèle à réponse univariée. Le seul critère réellement observable est la distribution des *résidus* du modèle (*i.e.* les écarts entre les valeurs réellement observées et celles prédites par le modèle). Pour que le modèle soit validé, les résidus doivent être distribués normalement.

Pour faire cette vérification : `plotresid(modele)`[1]. L'hypothèse de normalité est acceptée lorsque les points sont à peu près alignés sur une droite. Ces points peuvent ne pas être parfaitement alignés sur la droite, mais s'ils restent à peu près dans l'intervalle de confiance de celle-ci (représenté en pointillés), l'ajustement est considéré comme correct.

Test

L'effet des variables explicatives est testé par une MANOVA (*Multivariate ANalysis Of VAriance*), *via* `Manova(modele)`[2]. Le test réalisé est de type II (voir fiche **50** pour une explication détaillée des hypothèses testées). Plusieurs statistiques de test sont disponibles en MANOVA, celle utilisée ici est la statistique de Pillai - Bartlett.

Si un facteur, ou une interaction entre plusieurs facteurs, a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Pour les réaliser : `pairwise.manova(reponse,facteur)`[1] (où `facteur` peut être `facteur1:facteur2` si c'est une interaction qui a un effet significatif).

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur les variables à expliquer, mais également de prédire les valeurs que prendraient ces variables à expliquer pour des valeurs *connues* des variables explicatives. Prédire des moyennes (une par variable à expliquer) nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

- donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`
- créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- manova(reponse~facteur*covariable)
```

On peut prédire la valeur de chaque variable à expliquer de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10))
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)))
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau)
```

82. L'Analyse Discriminante Linéaire (LDA) – aspects inférentiels

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]MASS, [2]RVAideMemoire

Préparation des données

Les données doivent être contenues dans un tableau avec en colonnes les variables (avec un titre) et en lignes les individus. La 1ère case du tableau (en haut à gauche) doit être vide si la 1ère colonne contient le nom des individus. Donner un nom aux individus est conseillé pour faciliter l'interprétation de l'analyse. Pour des raisons pratiques, il est conseillé de mettre le facteur (*i.e.* la variable définissant les groupes à discriminer) avant les variables quantitatives.

Le tableau ne doit pas contenir de données manquantes (`NA`). Pour supprimer les lignes qui en possèdent une fois dans **R** : `tableau<-na.omit(tableau)`, où `tableau` est le tableau de départ.

Lancement de l'analyse

La LDA exige théoriquement que les variables explicatives (quantitatives) suivent une distribution normale multivariée et que leurs matrices de variance-covariance soient homogènes. On peut tester graphiquement la 1ère condition (voir fiche **44**), mais il n'existe pas réellement de test fiable pour la 2nde. En pratique la LDA est toutefois robuste au non-respect de ces deux conditions.

La LDA est créée *via* `LDA<-lda(explicatives,facteur)`[1] où `explicatives` correspond aux colonnes du tableau contenant les variables quantitatives et `facteur` est la colonne contenant le facteur. Par défaut tous les axes possibles sont retenus, *i.e.* $n-1$ où n est le nombre de classes du facteur.

— EXEMPLE(S) —

Si le tableau contient cinq colonnes et que le facteur est dans la 1ère :

```
> LDA <- lda(tableau[,2:5],tableau[,1])
```
[1]

Ou encore, si le facteur s'appelle `facteur` :

```
> LDA <- lda(tableau[,2:5],tableau$facteur)
```
[1]

Estimation de la qualité de la discrimination/prédiction

Il existe trois approches permettant d'estimer la qualité de la discrimination :

- les deux première répondent à la fois aux questions « À quel point le facteur est-il réellement discriminant ? » et « Avec quelle précision peut-on prédire le groupe d'un individus ? ». Les analyses discriminantes peuvent en effet avoir un objectif tant explicatif que prédictif. L'idée centrale des deux approches est de construire un modèle sur une partie des données, puis de prédire le groupe des individus n'étant pas intervenus dans la construction du modèle. On confronte alors les groupes prédits avec les groupes réellement observés, ce qui donne un indicateur de la qualité de la prédiction (ou de la discrimination) : le *pourcentage d'erreur de classification*. Plus ce pourcentage est faible, meilleure est la prédiction (ou discrimination). Les deux approches diffèrent dans leur façon d'estimer le pourcentage :
  - échantillon d'entraînement et échantillon de test : le jeu de données est divisé en deux parties. La première, l'*échantillon d'entraînement* (habituellement 2/3 des individus, choisis aléatoirement), sert à définir le modèle ; la seconde, l'*échantillon de test* (le 1/3 restant), sert à la prédiction. Il n'y a ici qu'un seul modèle de construit, et une estimation du pourcentage d'erreur de classification. Cette méthode est adaptée lorsque les individus sont nombreux (une centaine au minimum), ce qui permet d'avoir un échantillon d'entraînement assez grand pour estimer précisément le modèle et un échantillon de test assez grand pour estimer de manière fiable le pourcentage d'erreur de classification
  - validation croisée : le jeu de données est divisé aléatoirement en M groupes de même taille. On en utilise $M-1$ pour construire le modèle, et le dernier pour la prédiction. L'opération est répétée M fois, avec à chaque fois un groupe différent utilisé pour la prédiction. On obtient donc M pourcentages d'erreur de classification, dont la moyenne est utilisée comme indicateur global. L'ensemble du processus peut être répété n fois (avec à chaque fois une division différente

– puisqu'aléatoire – du jeu de données en M groupes) pour plus de fiabilité. Il y a donc ici M (voire $M \times n$) modèles de construits, et autant de pourcentages de calculés. La moyenne de tous ces pourcentages est utilisée comme indicateur global. Cette méthode est adaptée lorsque les individus ne sont pas très nombreux (moins d'une centaine)

– la troisième approche consiste en un test statistique, qui dira si les groupes sont significativement différents ou non. Cette méthode permet certes d'obtenir une *p-value*, mais elle apporte une information bien moins précise que les deux précédentes. Si elle est choisie, il est conseillé de l'accompagner d'un pourcentage d'erreur de classification.

Échantillon d'entraînement et échantillon de test

Tout le processus est réalisé *via* `DA.confusion(LDA)`[2]. La fonction choisit par défaut (et aléatoirement) 2/3 des individus ayant servi à construire `LDA` pour constituer l'échantillon d'entraînement. Cette proportion peut être modifiée grâce à l'argument `train=proportion` où `proportion` vaut entre 0 et 1.

Validation croisée

Tout le processus est réalisé *via* `DA.valid(LDA)`[2]. La fonction divise par défaut le jeu de données en 10 groupes et répète l'ensemble des opérations 20 fois. Ces valeurs peuvent être modifiées grâce aux arguments `M` et `nrep` respectivement.

Test

Le test utilisé est une analyse de variance multivariée ou MANOVA (paramétrique). Voir fiche **81** pour une explication détaillée de l'analyse.

Le modèle utilisé pour le test s'écrit : `modele<-manova(as.matrix(tableau[,colonnes])~facteur,data=tableau)`.

EXEMPLE(S)

Si le tableau contient cinq colonnes, que le facteur est contenu dans la 1ère et qu'il s'appelle `facteur`, la LDA est construite de cette façon :

```
> LDA <- lda(tableau[,2:5],tableau$facteur)[1]
```

Et le modèle pour le test de la suivante :

```
> modele <- manova(as.matrix(tableau[,2:5])~facteur,data=tableau)
```

Prédiction à partir du modèle

L'un des intérêt des analyses discriminantes est de pouvoir prédire le groupe d'un individu pour des valeurs *connues* des variables explicatives. Prédire un groupe nécessite donc de fixer la valeur de *toutes* les variables explicatives.

La prédiction se fait en deux étapes :

1. créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables de la LDA), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau)
2. réaliser la prédiction *via* `predict(LDA,newdata=tableau)$class` où `tableau` est le tableau créé à l'étape précédente.

83. La régression PLS discriminante (PLS-DA) – aspects inférentiels

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`mixOmics`, [2]`RVAideMemoire`

Préparation des données

Les données doivent être contenues dans un tableau avec en colonnes les variables (avec un titre) et en lignes les individus. La 1ère case du tableau (en haut à gauche) doit être vide si la 1ère colonne contient le nom des individus. Donner un nom aux individus est conseillé pour faciliter l'interprétation de l'analyse. Pour des raisons pratiques, il est conseillé de mettre le facteur (*i.e.* la variable définissant les groupes à discriminer) avant les variables quantitatives.

Lancement de l'analyse

La PLS-DA est créée *via* `PLSDA<-plsda(explicatives,facteur)`[1] où `explicatives` correspond aux colonnes du tableau contenant les variables quantitatives et `facteur` est la colonne contenant le facteur. Par défaut deux axes sont retenus, mais cette valeur peut être changée en utilisant l'argument `ncomp=k` où `k` est le nombre d'axes à retenir. En pratique on retient souvent $n-1$ axes, où n est le nombre de classes du facteur. On peut toutefois parfaitement en retenir moins pour faciliter l'interprétation (moyennant, forcément, une perte d'information).

— EXEMPLE(S) —

Si le tableau contient 25 colonnes et que le facteur est dans la 1ère :

```
> PLSDA <- plsda(tableau[,2:25],tableau[,1])[1]
```

Ou encore, si le facteur s'appelle `facteur` :

```
> PLSDA <- plsda(tableau[,2:25],tableau$facteur)[1]
```

Estimation de la qualité de la discrimination/prédiction

Il existe trois approches permettant d'estimer la qualité de la discrimination :

- les deux première répondent à la fois aux questions « À quel point le facteur est-il réellement discriminant ? » et « Avec quelle précision peut-on prédire le groupe d'un individus ? ». Les analyses discriminantes peuvent en effet avoir un objectif tant explicatif que prédictif. L'idée centrale des deux approches est de construire un modèle sur une partie des données, puis de prédire le groupe des individus n'étant pas intervenus dans la construction du modèle. On confronte alors les groupes prédits avec les groupes réellement observés, ce qui donne un indicateur de la qualité de la prédiction (ou de la discrimination) : le *pourcentage d'erreur de classification*. Plus ce pourcentage est faible, meilleure est la prédiction (ou discrimination). Les deux approches diffèrent dans leur façon d'estimer le pourcentage :
  - échantillon d'entraînement et échantillon de test : le jeu de données est divisé en deux parties. La première, l'*échantillon d'entraînement* (habituellement 2/3 des individus, choisis aléatoirement), sert à définir le modèle ; la seconde, l'*échantillon de test* (le 1/3 restant), sert à la prédiction. Il n'y a ici qu'un seul modèle de construit, et une estimation du pourcentage d'erreur de classification. Cette méthode est adaptée lorsque les individus sont nombreux (une centaine au minimum), ce qui permet d'avoir un échantillon d'entraînement assez grand pour estimer précisément le modèle et un échantillon de test assez grand pour estimer de manière fiable le pourcentage d'erreur de classification
  - validation croisée : le jeu de données est divisé aléatoirement en M groupes de même taille. On en utilise $M-1$ pour construire le modèle, et le dernier pour la prédiction. L'opération est répétée M fois, avec à chaque fois un groupe différent utilisé pour la prédiction. On obtient donc M pourcentages d'erreur de classification, dont la moyenne est utilisée comme indicateur global. L'ensemble du processus peut être répété n fois (avec à chaque fois une division différente – puisqu'aléatoire – du jeu de données en M groupes) pour plus de fiabilité. Il y a donc ici M (voire $M \times n$) modèles de construits, et autant de pourcentages de calculés. La moyenne de tous ces pourcentages est utilisée comme indicateur global. Cette méthode est adaptée lorsque les individus ne sont pas très nombreux (moins d'une centaine)

– la troisième approche consiste en un test statistique, qui dira si les groupes sont significativement différents ou non. Cette méthode permet certes d'obtenir une *p-value*, mais elle apporte une information bien moins précise que les deux précédentes. Si elle est choisie, il est conseillé de l'accompagner d'un pourcentage d'erreur de classification.

Échantillon d'entraînement et échantillon de test

Tout le processus est réalisé *via* `DA.confusion(PLSDA)`[2]. La fonction choisit par défaut (et aléatoirement) 2/3 des individus ayant servi à construire `PLSDA` pour constituer l'échantillon d'entraînement. Cette proportion peut être modifiée grâce à l'argument `train=proportion` où `proportion` vaut entre 0 et 1.

Validation croisée

Tout le processus est réalisé *via* `DA.valid(PLSDA)`[2]. La fonction divise par défaut le jeu de données en 10 groupes et répète l'ensemble des opérations 20 fois. Ces valeurs peuvent être modifiées grâce aux arguments `M` et `nrep` respectivement.

Test

Le test utilisé est un test non paramétrique (par permutation) sans nom particulier, basé sur le pourcentage d'erreur de classficiation de `PLSDA` (calculé par validation croisée). Il est réalisé *via* `PLSDA.test(PLSDA)`[2].

Remarque : la procédure est longue car basée sur environ 1000 fois la même opération, et prend plusieurs minutes pour être réalisée.

Prédiction à partir du modèle

L'un des intérêt des analyses discriminantes est de pouvoir prédire le groupe d'un individu pour des valeurs *connues* des variables explicatives. Prédire un groupe nécessite donc de fixer la valeur de *toutes* les variables explicatives.

La prédiction se fait en trois étapes :

1. créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables de la PLS-DA), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau)
2. réaliser la prédiction *via* `prediction<-predict(PLSDA,newdata=tableau)$class[[3]]` où `tableau` est le tableau créé à l'étape précédente. Le résultat renvoyé par la fonction n'est pas directement le(s) groupe(s) prédit(s) mais un tableau de chiffres
3. retrouver les groupes correspondant à ces chiffres : `levels(tableau$facteur[prediction[,ncol(prediction)]])`.

Bibliographie et ouvrages/documents/liens recommandés

Livres

Crawley M.J. (2007) The R Book. Éditions John Wiley & Sons, inc.

Dagnelie P. (2006a) Statistique théorique et appliquée. 1. Statistique descriptive et base de l'inférence statistique. 3ème édition. Éditions De Boeck.

Dagnelie P. (2006b) Statistique théorique et appliquée. 2. Inférence statistique à une et à deux dimensions. 3ème édition. Éditions De Boeck.

Millot G. (2008) Comprendre et réaliser les tests statistiques à l'aide de R. 2ème édition. Éditions De Boeck.

Documents en ligne

Bates D. (2010) lme4 :Mixed-effects modeling with R. [*http ://lme4.r-forge.r-project.org*]

Champely S. (2005) Introduction à l'analyse multivariée (factorielle) sous R. [*http ://pierre.jeandenand.free.fr/Cours/Statistiques%20avec%20Rgui.pdf*]

Dagnelie P. (2003) Principes d'expérimentation : planification des expériences et analyse de leurs résultats. Les presses agronomiques de Gembloux. [*http ://www.dagnelie.be*]

Fox J. (2002) Cox Proportional-Hazards Regression fot Survival Data. [*http ://cran.r-project.org*, rubrique *Contributed* : *Web Appendix to the book "An R and S-PLUS Companion to Applied Regression"*]

Paradis E. (2002) R pour les débutants. [*http ://cran.r-project.org*, rubrique *Contributed*]

Poinsot D. (2004) Statistiques pour statophobes. [*http ://perso.univ-rennes1.fr/denis.poinsot*]

Poinsot D. (2005) R pour les statophobes. [*http ://perso.univ-rennes1.fr/denis.poinsot*]

Les nombreux cours en ligne de **D. Chessel**, **A.B. Dufour**, **J.R. Lobry** et **M. Royer** : *http ://pbil.univ-lyon1.fr/R/enseignement.html*

Les cours en ligne de **M-L. Delignette-Muller** : *http ://www2.vet-lyon.fr/ens/biostat/accueil.html*

Le site de R : *http ://www.R-project.org*.

Groupes d'utilisateurs

Le forum du groupe des utilisateurs du logiciel R : *http ://forums.cirad.fr/logiciel-R/index.php*

Semin-R, un autre groupe d'utilisateurs de R : *http ://rug.mnhn.fr/semin-r*